SPERRY

PUBLICATIONS UPDATE

Operating System/3 (OS/3)

MAPPER™ 80 Forms Generation and Utilities User Guide

UP-9736-A

This Library Memo announces the release and availability of "Updating Package A to SPERRY Operating System/3 (OS/3) MAPPER 80 Forms Generation and Utilities User Guide", UP-9736.

MAPPER 80 is a general online report processing system that uses a report structured data base. You do not have to understand programming to use MAPPER 80. Users can create and delete reports; manipulate data within reports; design new formats and applications; execute runs; and design new runs.

This manual tells how to generate report forms and create reports. It also discusses MAPPER 80 utilities and their use with or without your own code, including:

- MAPLOD – creation of reports
- MAPLST – report print control
- MAPRAD – creation of data or run reports and addition or appending of reports from MIRAM files to the data base
- MAPDMS – subroutine to your own program to initiate, manipulate, and terminate reports offline when repetitive functions are performed on reports

This update provides additional clarification for the FGEN function, and gives expanded guidelines for specifying a password. In addition, the user-id name in the sample MAPRAD control statement is changed from JDOE to MAPR.

Copies of Updating Package A are now available for requisitioning. Either the updating package only or the complete manual with the updating package may be requisitioned by your local Sperry representative. To receive only the updating package, order UP-9736-A. To receive the complete manual, order UP-9736.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ, CZ, MZ, 28U, 29U,	Mailing Lists B00 and B36 (Package A to UP-9736, Cover and 31 pages plus Memo)	Library Memo for UP-9736–A
		RELEASE DATE: September, 1984

SPERRY

```
##
I#      ATTN: LIBRARIAN
                   2 COPIES
        00636
        CA1777M20164  UP  9736

        SPERRY                         3        UAM
        BRANCH LIBRARY
        1818 CORNWALL ST SUITE      U
        VANCOUVER BC  V6J 1C7
                                             CAV
```

PUBLICATIONS RELEASE

Operating System/3 (OS/3)

MAPPER® 80 Forms Generation and Utilities

User Guide

UP-9736

This Library Memo announces the release and availability of "SPERRY® Operating System/3 (OS/3) MAPPER 80 Forms Generation and Utilities User Guide", UP-9736.

The MAPPER 80 software is a general online report processing system that uses a report structured data base. You do not have to understand programming to use the MAPPER 80 software. Users can create and delete reports; manipulate data within reports; design new formats and applications; execute runs; and design new runs.

MAPPER 80 capabilities include column-formed report entry, storage, retrieval, updating, and hard-copy output. On column-formed reports, report processing functions include searching, sorting, matching, totalizing, and character-string location and change.

Run functions automate and speed report processing when manual functions are combined and used repetively.

This manual tells how to generate report forms and create reports. It also discusses MAPPER 80 utilities and their use, with or without your own code, including:

- MAPLOD – creation of reports
- MAPLST – report print control
- MAPRAD – creation of data or run reports and addition or appending of reports from MIRAM files to the data base
- MAPDMS – subroutine to your own program to initiate, manipulate, and terminate reports offline when repetitive functions are performed on reports

This user guide is one of a series of MAPPER 80 manuals designed to instruct both novice and experienced users in the MAPPER 80 system. Current versions of the following MAPPER 80 manuals are also available:

- Introduction, UP-10000
- Operator and coordinator user guide, UP-9737
- Manual functions user guide, UP-9735
- Run functions user guide, UP-9734

Additional copies may be ordered by your local Sperry representative.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ, CZ and MZ	Mailing Lists B00, B36, 28U, and 29U (Cover and 164 pages)	Library Memo for UP-9736
		RELEASE DATE: December, 1983

SPERRY

PUBLICATIONS RELEASE

Operating System/3 (OS/3)

MAPPER® 80 Forms Generation and Utilities User Guide

UP-9736

This Library Memo announces the release and availability of "SPERRY® Operating System/3 (OS/3) MAPPER 80 Forms Generation and Utilities User Guide", UP-9736.

The MAPPER 80 software is a general online report processing system that uses a report structured data base. You do not have to understand programming to use the MAPPER 80 software. Users can create and delete reports; manipulate data within reports; design new formats and applications; execute runs; and design new runs.

MAPPER 80 capabilities include column-formed report entry, storage, retrieval, updating, and hard-copy output. On column-formed reports, report processing functions include searching, sorting, matching, totalizing, and character-string location and change.

Run functions automate and speed report processing when manual functions are combined and used repetively.

This manual tells how to generate report forms and create reports. It also discusses MAPPER 80 utilities and their use, with or without your own code, including:

- MAPLOD – creation of reports
- MAPLST – report print control
- MAPRAD – creation of data or run reports and addition or appending of reports from MIRAM files to the data base
- MAPDMS – subroutine to your own program to initiate, manipulate, and terminate reports offline when repetitive functions are performed on reports

This user guide is one of a series of MAPPER 80 manuals designed to instruct both novice and experienced users in the MAPPER 80 system. Current versions of the following MAPPER 80 manuals are also available:

- Introduction, UP-10000
- Operator and coordinator user guide, UP-9737
- Manual functions user guide, UP-9735
- Run functions user guide, UP-9734

Additional copies may be ordered by your local Sperry representative.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS:
Mailing Lists BZ, CZ and MZ	Mailing Lists B00, B36, 28U, and 29U (Cover and 164 pages)	Library Memo for UP-9736
		RELEASE DATE: December, 1983

MAPPER 80
Forms Generation and Utilities

OS/3

User Guide

SPERRY

H

UP-9736

PAGE STATUS SUMMARY

ISSUE: Update A – UP-9736
RELEASE LEVEL: 8.1 Forward

Part/Section	Page Number	Update Level
Cover/Disclaimer		A
PSS	1	A
Preface	1	A
	2	Orig.
Contents	1 thru 4	Orig.
1	1	Orig.
2	1 thru 7	Orig.
	8	A
	9 thru 11	Orig.
	12	A
	13 thru 22	Orig.
3	1 thru 7	Orig.
	8	A
	9 thru 19	Orig.
	20	A
	21 thru 28	Orig.
4	1 thru 15	Orig.
5	1 thru 3	Orig.
	4	A
	5	Orig.
6	1 thru 3	Orig.
	4, 5	A
	6, 7	Orig.
	8	A
	9	Orig.
	10	A
	11	Orig.
	12	A
	13	Orig.
	14, 15	A
	16 thru 22	Orig.
	23	A
	24, 25	Orig.
	26	A
	27, 28	Orig.
Appendix A	1 thru 49	Orig.
Index	1 thru 7	Orig.
User Comment Sheet		

Part/Section	Page Number	Update Level

Part/Section	Page Number	Update Level

All the technical changes are denoted by an arrow (⟹) in the margin. A downward pointing arrow (⇓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (⇑) is found. A horizontal arrow (⟹) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

Preface

This manual is one of a series designed to instruct and guide the SPERRY Operating System/3 (OS/3) MAPPER 80 system or Series 90 user. It specifically describes the:

- MAPPER 80 report load utility (MAPLOD)
- MAPPER 80 report print utility (MAPLST)
- MAPPER 80 report add utility (MAPRAD)
- non-MAPPER 80 file or data base report utility (MAPDMS)

Current versions of four other MAPPER 80 manuals are also available:

- Introduction, UP-10000

  Provides an overview of MAPPER 80 system.

- Operator and coordinator user guide, UP-9737

  Discusses MAPPER 80 operations, including system generation and data base system coordination.

- Manual functions user guide, UP-9735

  Describes all the manual functions and how they manipulate reports or data within reports.

- Run functions user guide, UP-9734

  Describes a powerful set of procedure-like run functions that call sequences of manual functions to manipulate reports or the data within them.

This manual is divided into six sections:

- Section 1. Introduction

  Provides an overview of the four MAPPER 80 utilities described in this manual.

- Section 2. Forms Generation

  Describes the steps needed to define a new report type. Use this section only when defining a report type for the first time.

- Section 3. MAPLOD Utility

  Tells how to use the MAPLOD utility to create reports on a data base from the contents of an OS/3 spool file and how to include your own code.

- Section 4. MAPLST Utility

  Describes how to use the MAPLST utility to start, stop, and control the printing of reports.

- Section 5. MAPRAD Utility

  Describes how to use the MAPRAD utility to create data, run reports, and add reports to existing reports from a MIRAM file to a MAPPER 80 data base.

- Section 6. MAPDMS Utility

  Describes the MAPDMS utility; the CALL functions that initiate and terminate the MAPDMS utility and reports; the processing report line functions that allow you to change or manipulate one line of a particular report; and the rollback and lock procedures. This section is for advanced programmers who write their own programs to interface the MAPPER 80 data base.

In addition, Appendix A provides a list of MAPPER 80 error messages.

Contents

1. Introduction

This manual explains how to use MAPPER 80 software to:

- generate report forms;
- create and print reports using the MAPLOD and MAPLST utilities;
- add reports to the MAPPER 80 data base using the MAPRAD utility; and
- use your own programs to directly query and update MAPPER 80 reports.

A knowledge of SPERRY Operating System/3 (OS/3) and job control streams is required to use these utilities, and advanced programming techniques are necessary to properly use the MAPDMS utility.

The MAPLOD and MAPLST utilities perform complementary operations. MAPLOD allows you to enter MAPPER 80 reports into the data base from an OS/3 spool file. MAPLST produces MAPPER 80 reports on the system printer and allows you to control print operations (i.e., line spacing and headers).

The MAPRAD utility is a batch utility that adds reports to the MAPPER 80 data base from a MIRAM file.

The MAPDMS utility acts as a subroutine to your own program allowing you to access and manipulate a MAPPER 80 data base at a more intricate level.

2. Forms Generation

2.1. HOW TO GENERATE A FORM

To enter data into the MAPPER 80 data base, it is first necessary to define a report form type.

The MAPPER 80 data base consists of mode, type, and report identification (RID). Within the data base, all reports in a type within a mode have the same data format. The format of each type is defined in RID 0 when new data is stored.

The basic responsibility for forms generation lies with the MAPPER 80 coordinator. When it is necessary to generate a report form, apply to the coordinator.

Figure 2–1 illustrates the process of forms generation.

- Estimate of report and service

  When a report is being newly created, the user clarifies the report's objective and service. In addition, he considers and justifies the frequency of access to the report and cost of operation.

- Application to the coordinator

  The person who wants to add a new report in the MAPPER 80 data base must apply for a new report using, for example, a MAPPER 80 Use Service Registration Application Form to register the report. This application must then be checked and approved by the coordinator. It is the coordinator's responsibility to maintain the performance efficiency of the MAPPER 80 software. Figure 2–2 shows a sample report application form.

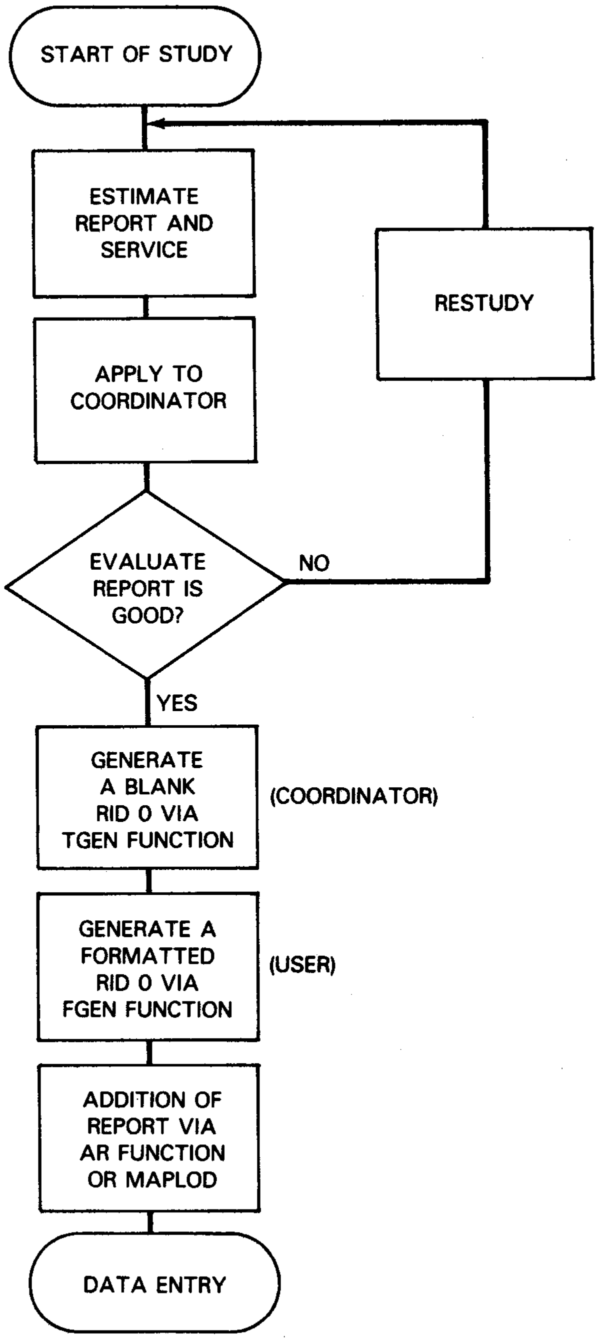
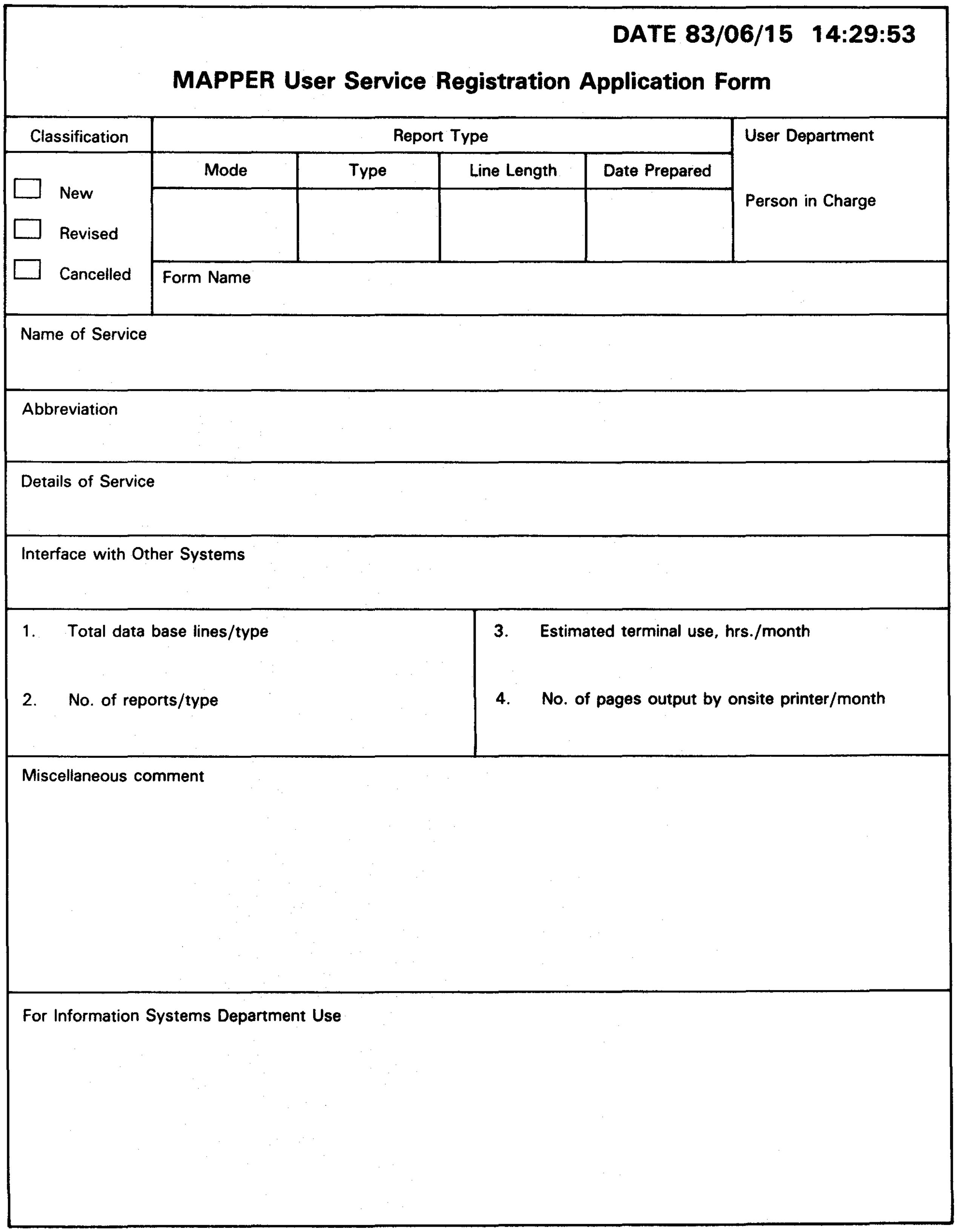

START OF STUDY
ESTIMATE REPORT AND SERVICE
APPLY TO COORDINATOR
RESTUDY
EVALUATE REPORT IS GOOD?
NO
YES
GENERATE A BLANK RID 0 VIA TGEN FUNCTION
(COORDINATOR)
GENERATE A FORMATTED RID 0 VIA FGEN FUNCTION
(USER)
ADDITION OF REPORT VIA AR FUNCTION OR MAPLOD
DATA ENTRY

Figure 2–1. MAPPER 80 Forms Generation Process



DATE 83/06/15 14:29:53

MAPPER User Service Registration Application Form

Classification	Report Type				User Department
	Mode	Type	Line Length	Date Prepared	
☐ New ☐ Revised ☐ Cancelled					Person in Charge
	Form Name				

Name of Service

Abbreviation

Details of Service

Interface with Other Systems

1. Total data base lines/type	3. Estimated terminal use, hrs./month
2. No. of reports/type	4. No. of pages output by onsite printer/month

Miscellaneous comment

For Information Systems Department Use

Figure 2–2. Sample Application Form

- Evaluation of report by coordinator

  Based on the application, the coordinator evaluates the following:

  1. Volume of the report and its effect on the data base
  2. Frequency of access to the report for use and updating
  3. General application and special characteristics of the report, and the existence of similar reports
  4. Type of terminal and periods of use
  5. Miscellaneous special conditions

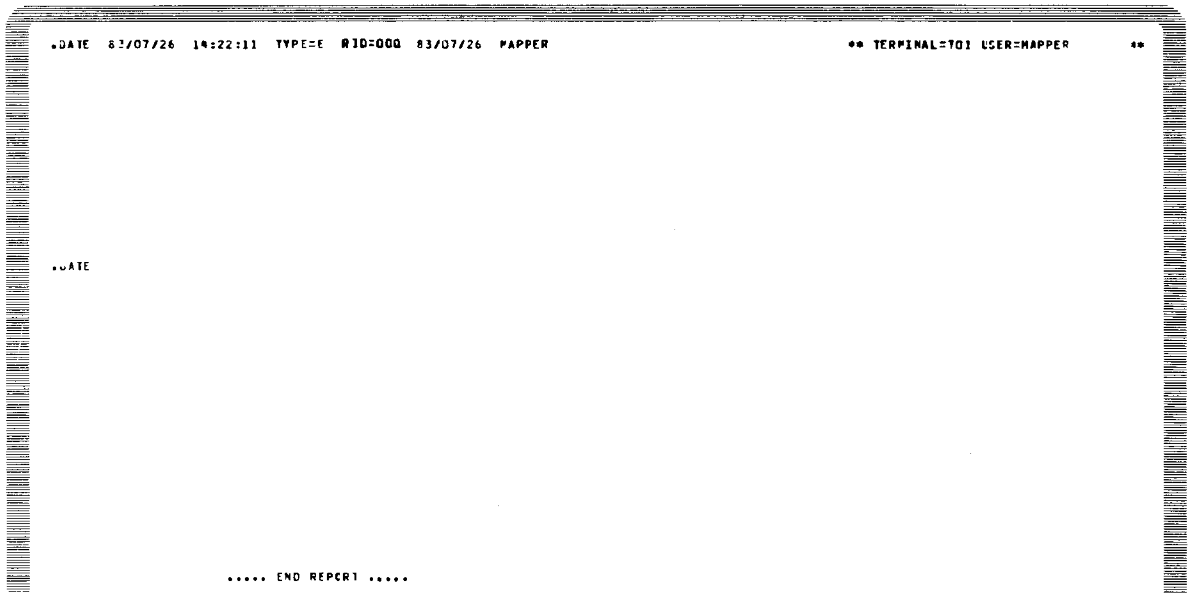
  If the coordinator accepts the report, he generates a new form type. When the new report is inappropriate, you must make a further study. Depending on the priority of the report, the coordinator may give conditional approval.

  NOTE:

  For convenience in the following forms generation description, we show screens 132 characters wide and 40 lines long. When RIDs are more than 80 columns wide or more than 20 lines long, you must use the shift or roll functions to display the additional columns or lines.

- Generation of a blank RID 0

  The coordinator generates a blank RID 0 for the newly allocated form type, and informs you of it. A blank RID 0 is a RID 0 form whose contents are blank. Figure 2–3 shows a sample blank RID 0. The coordinator uses the TGEN function to register the form name, line length, mode, password, and user-id on the data base.

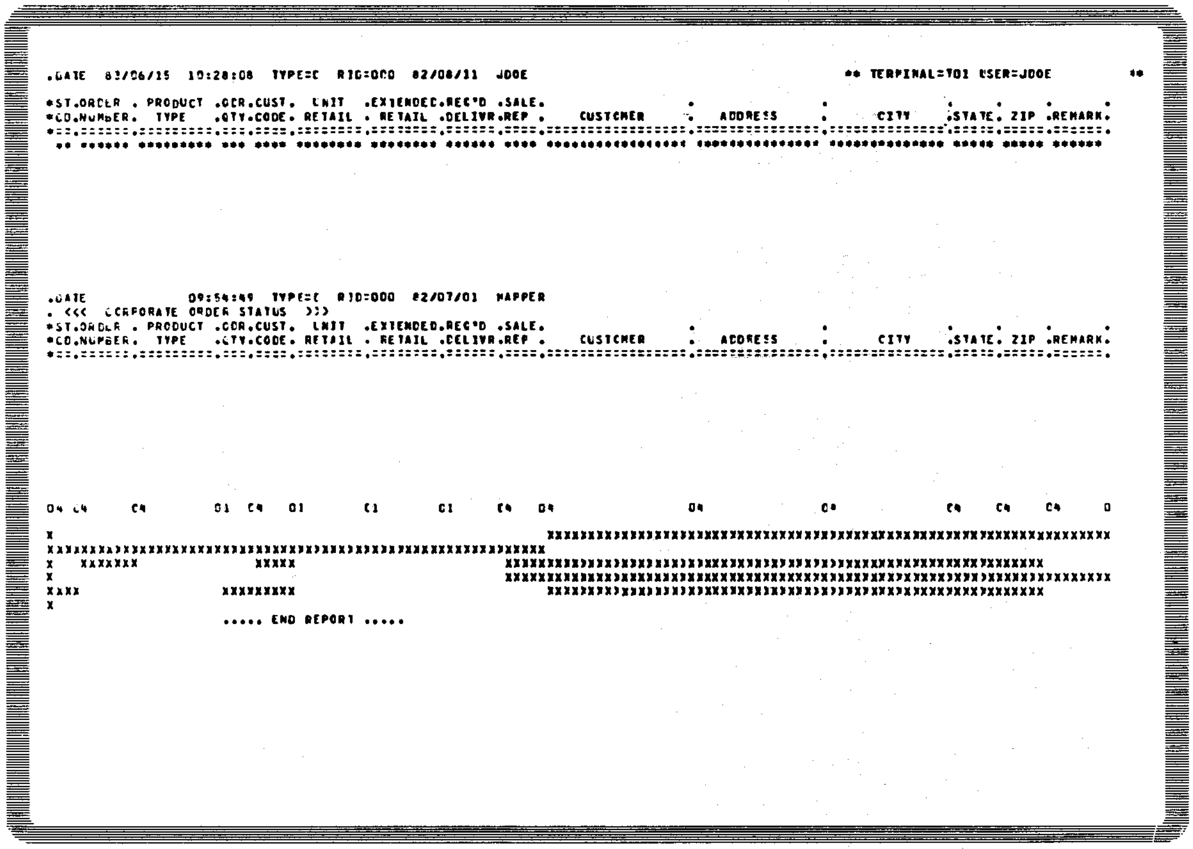


```
.DATE  83/07/26  14:22:11  TYPE=E  RID=000  83/07/26  MAPPER                                    ** TERMINAL=T01 USER=MAPPER        **















.DATE






















                  ..... END REPORT .....
```

Figure 2–3. Sample Blank RID 0 Form

- Generation of a formatted RID 0

  Using the blank RID 0 generated by the coordinator, you generate a formatted RID 0 via the FGEN function and line update functions. Figure 2–4 shows an example of a formatted RID 0.



```
.DATE  83/06/15  10:28:08  TYPE=C  RID=000  82/08/11  JDOE                   ** TERMINAL=T01 USER=JDOE          **
*ST.ORDER . PRODUCT .ORR.CUST.  UNIT  .EXTENDED.REC'D .SALE.                 .                .               .              .     .     .      .
*CO.NUMBER.  TYPE   .QTY.CODE. RETAIL . RETAIL .DELIVR.REP .    CUSTOMER     .     ADDRESS    .      CITY     .STATE. ZIP .REMARK.
*==.======.=========.===.====.========.========.======.====.=================.================.===============.=====.=====.======.
 ** ****** ********* *** **** ******** ******** ****** **** ***************** **************** *************** ***** ***** ******










.DATE               09:54:49  TYPE=C  RID=000  82/07/01  MAPPER
. <<<  CORPORATE ORDER STATUS  >>>
*ST.ORDER . PRODUCT .COR.CUST.  UNIT  .EXTENDED.REC'D .SALE.                 .                .               .              .     .     .      .
*CO.NUMBER.  TYPE   .QTY.CODE. RETAIL . RETAIL .DELIVR.REP .    CUSTOMER     .     ADDRESS    .      CITY     .STATE. ZIP .REMARK.
*==.======.=========.===.====.========.========.======.====.=================.================.===============.=====.=====.======.










04 04      C4          01  C4   01         C1        C1    C4   D4              D4            C4            C4    C4    C4     0

X                                                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X   XXXXXXX                XXXXX                      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X                                                     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXX                   XXXXXXXXX                           XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X
                       ..... END REPORT .....
```

Figure 2–4. Sample RID 0 Containing Information

- Addition of a report

  On completion of the formatted RID 0, you add reports in the particular RID type by using the AR function or the MAPLOD utility. (See Section 3.) At this point, the report is ready for data entry.

2.2. RID 0

RID 0 is fixed at 40 lines and contains the following information:

- Line 1 (first line)

  Contains the date and time when the report was created and the user's identity.

- Mask (lines 2–16)

  Contains the mask images that receive parameters for using the SEARCH, MATCH, TOTALIZE, and other manual functions.

  Line 2 is blank. Lines 3 and 4 are header lines and have the same contents as the header lines in the basic report (lines 19 and 20). They define the fields. Line 5 is a dividing line (*=) that divides the headers from the data. The equal signs define each field and periods define the boundary between fields. A period is in the last column.

  Line 6 indicates the number of columns in each field by using asterisks. The asterisks correspond to the positions of the column in each field, while the boundaries of the fields are left blank. Lines 7 through 16 provide 10 blank tab lines with empty fields, each line having tabs in the field boundaries. These empty field boundaries receive parameters when the mask is displayed on the screen.

- Basic report (lines 17–31)

  Contains the basic format of the report type as added to the RID type via the ADD REPORT manual function. The basic report format consists of 15 lines plus 1 END REPORT line.

  Line 17 begins with .DATE for future use. Line 18 starts with a period and contains the title of the report. Lines beginning with asterisks (*) or periods (.) contain no tab characters and must be predefined. Lines 19 and 20 are header lines that define the fields, and line 21 is a dividing line. These three lines have the same content as lines 3, 4, and 5 in the mask. You can use the header lines with the MAPLOD utility.

  Line 22 is a blank tab line. Lines 23 through 31 are predefined lines where you enter data in the fields for future reference. The blank tab line and predefined blank lines (23 through 31) allow you to add lines to the report by using the ADD LINE manual function. Since these lines are set the same as the blank tab line, when you provide optional data in the predefined blank lines, you must set the tab in the last column of the tab line.

- Edit code (line 32)

  This line defines the character set type for the fields within the report lines.

  Specify an edit code for each field. When you update data with the UPDATE LINE manual function or check data while producing a report with the MAPLOD utility, the edit code verifies the characters used in each field.

  The first column in each field specifies the edit code, and a column corresponding to a tab line is assigned a zero. Table 2–1 shows edit codes used to verify fields in RID 0.

Table 2–1. Edit Codes Used in RID Field Verification

Edit Code	Meaning	Field Content Description
0	Tab column	Contains zero. Cannot be indicated in two consecutive fields
1	Numerical items	Contains numeric values, plus or minus signs before data, and decimal points within numeric data
2	Edit items with numeric values	Contains dollar signs ($) before data, commas, and decimal points within numeric data
4	Characters	Contains blanks, alphanumeric characters, Katakana characters, and special characters
5	Characters	Contains the same as edit code 4, except blank columns are not allowed

Edit codes 1 and 2 are right-justified fields, while all other codes are left-justified. Enter data anywhere in the field; it is automatically adjusted according to the edit code of the receiving field. However, the results of calculations performed by the TOTALIZE manual function are always right-justified when displayed in a field, regardless of the edit code.

- Blank (line 33)

  Blank line

- Format control lines (line 34–39)

  Defines up to six modifications of a particular basic format. For each modified format, specify all necessary fields by entering Xs in all pertinent columns. (See Figure 2–4.)

- END REPORT line (line 40)

The blank RID 0 generated by the coordinator contains only lines 1, 17, and 40. You add the other lines by using line update functions or FGEN function request.

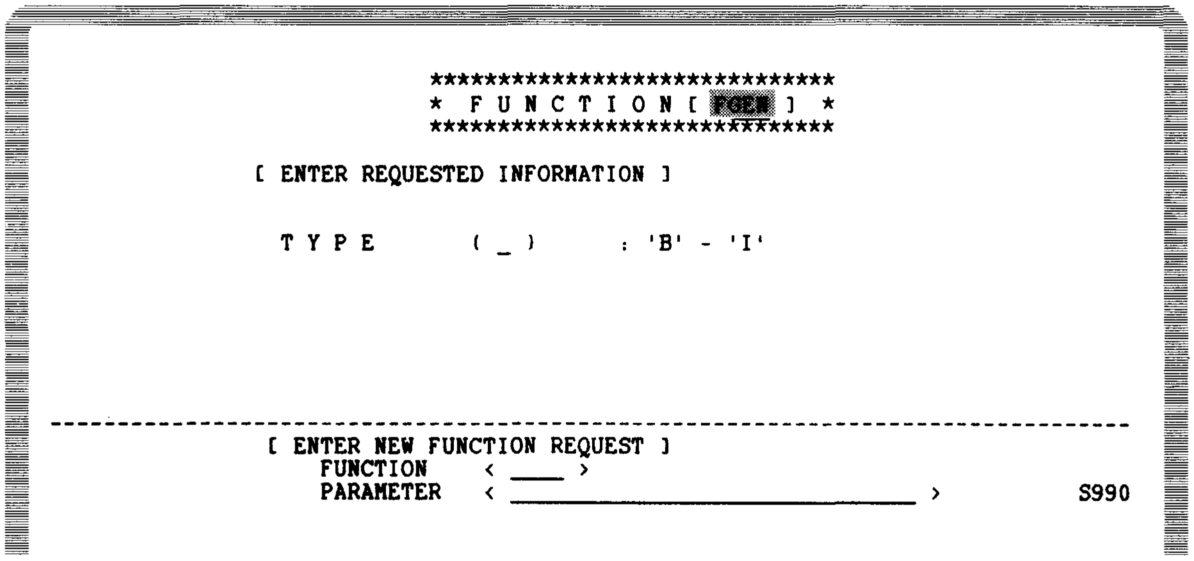
2.3. USING THE FGEN FUNCTION TO PREPARE RID 0 FOR DATA REPORTS

The coordinator generates the blank RID 0. You use the FGEN function to identify the RID type in which you want to create a report. (See Figure 2–4.) This function allows you to prepare headers and modified formats.

NOTE:

The FGEN function is invalid for modes 994, 996, and 998. This applies only to a MAPPER 80 coordinator since only a coordinator may access these modes. See the MAPPER 80 Operator and Coordinator User Guide, UP-9737 (current version).

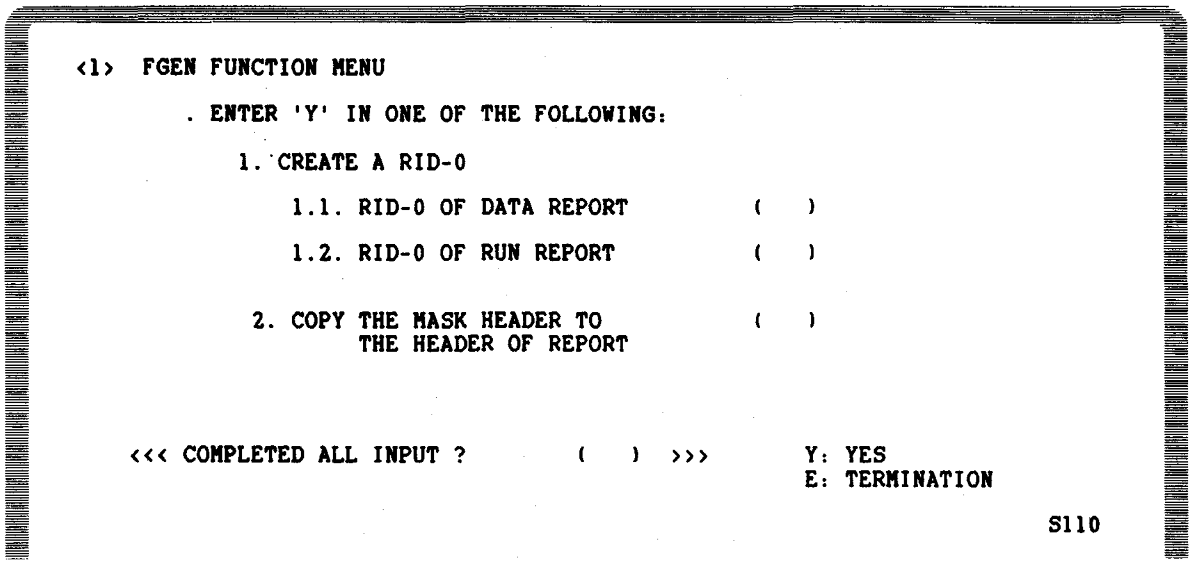
Using the interactive function, key in FGEN as the function name and transmit. Figure 2–5 shows the resulting FGEN screen:



```
                    ******************************
                    *  F U N C T I O N [ FGEN ]  *
                    ******************************

           [ ENTER REQUESTED INFORMATION ]

            T Y P E      ( _ )      : 'B' - 'I'



-------------------------------------------------------------------------
            [ ENTER NEW FUNCTION REQUEST ]
               FUNCTION     < ____ >
               PARAMETER    < ______________________________ >          S990
```

Figure 2–5. Function Request Screen Display

Enter the type name (B through I) in the TYPE field and transmit. Figure 2–6 shows the resulting menu screen that asks which FGEN option you want.



```
<1> FGEN FUNCTION MENU

          . ENTER 'Y' IN ONE OF THE FOLLOWING:

             1. CREATE A RID-0

                 1.1. RID-0 OF DATA REPORT          (   )

                 1.2. RID-0 OF RUN REPORT           (   )

              2. COPY THE MASK HEADER TO            (   )
                      THE HEADER OF REPORT



    <<< COMPLETED ALL INPUT ?       (   ) >>>      Y: YES
                                                   E: TERMINATION

                                                                 S110
```

Figure 2–6. FGEN Function Menu Screen Display

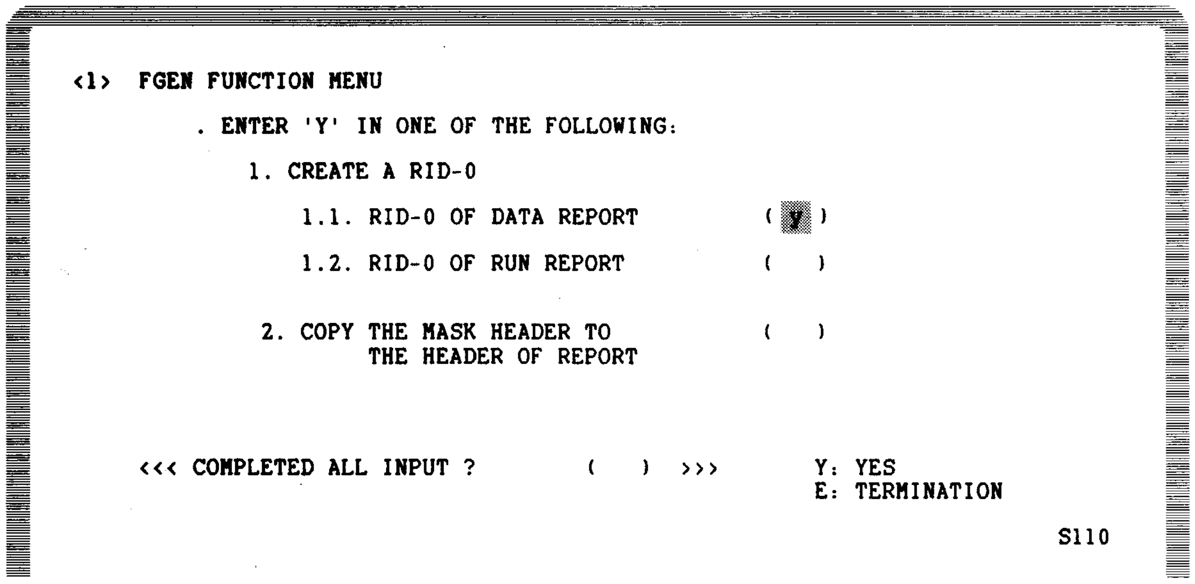
Choose from three different operations within the FGEN function:

1. Prepare RID 0 for a data report.
2. Prepare RID 0 for a run report.
3. Copy headers from the header section of the mask area to the header section of a basic report area. Use this function also when changing the content of header fields.

When preparing RID 0, be sure there are no reports, other than RID 0, in the form type. This is necessary because the new basic format of RID 0 does not consider report formats previously generated in this form type.

You can modify the established RID 0 as often as you want when there are no other reports in the form type.

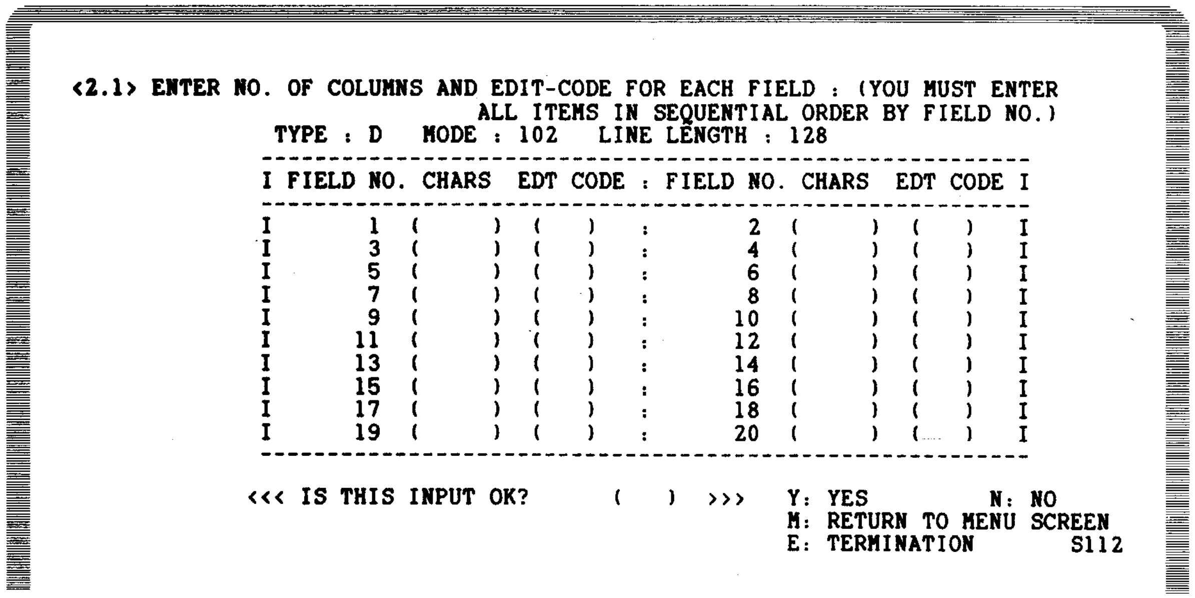
Figure 2–7 shows the selection of RID 0 OF DATA REPORT option. Note that in this example, the last field on the screen is not specified. When left blank, the default is Y.



```
<1>  FGEN FUNCTION MENU

          . ENTER 'Y' IN ONE OF THE FOLLOWING:

              1. CREATE A RID-0

                  1.1. RID-0 OF DATA REPORT          ( Y )

                  1.2. RID-0 OF RUN REPORT           (   )


               2. COPY THE MASK HEADER TO            (   )
                       THE HEADER OF REPORT



     <<< COMPLETED ALL INPUT ?         (   )  >>>        Y: YES
                                                      E: TERMINATION

                                                                     S110
```

Figure 2–7. FGEN Function Menu Screen with RID 0 Data Entry Choice

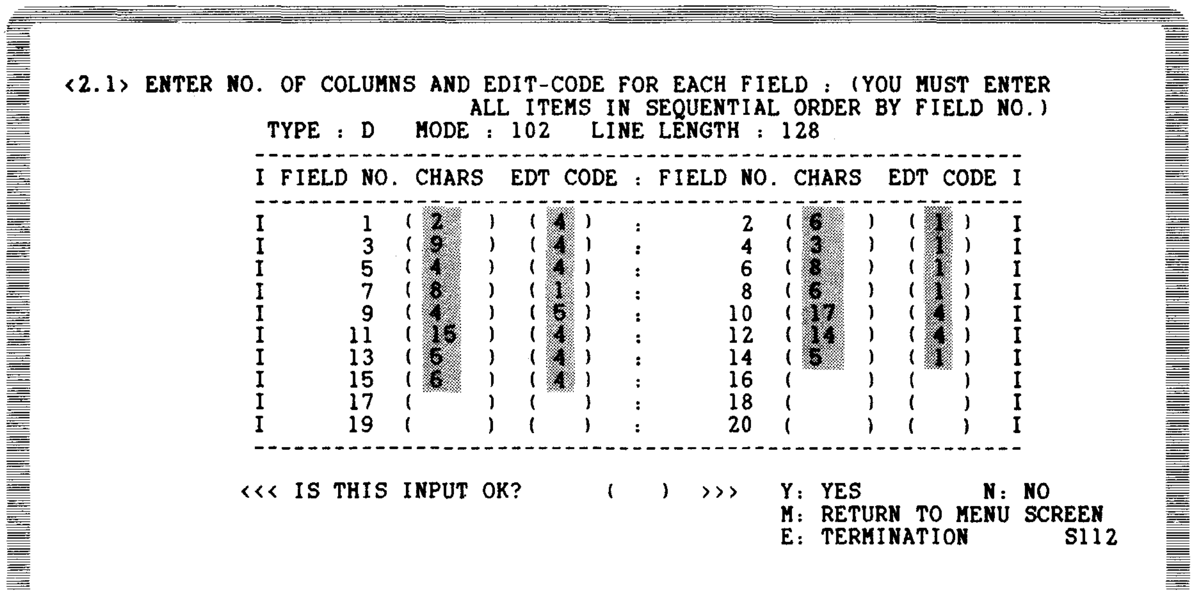
Figure 2–8 shows the data report format screen that results when you press the transmit key.



```
<2.1> ENTER NO. OF COLUMNS AND EDIT-CODE FOR EACH FIELD : (YOU MUST ENTER
                                ALL ITEMS IN SEQUENTIAL ORDER BY FIELD NO.)
              TYPE : D   MODE : 102   LINE LENGTH : 128
             ---------------------------------------------------------
             I FIELD NO. CHARS   EDT CODE : FIELD NO. CHARS   EDT CODE I
             ---------------------------------------------------------
             I        1  (     ) (   )    :        2  (     ) (   )   I
             I        3  (     ) (   )    :        4  (     ) (   )   I
             I        5  (     ) (   )    :        6  (     ) (   )   I
             I        7  (     ) (   )    :        8  (     ) (   )   I
             I        9  (     ) (   )    :       10  (     ) (   )   I
             I       11  (     ) (   )    :       12  (     ) (   )   I
             I       13  (     ) (   )    :       14  (     ) (   )   I
             I       15  (     ) (   )    :       16  (     ) (   )   I
             I       17  (     ) (   )    :       18  (     ) (   )   I
             I       19  (     ) (   )    :       20  (     ) (   )   I
             ---------------------------------------------------------

             <<< IS THIS INPUT OK?     (   )  >>>  Y: YES        N: NO
                                                  M: RETURN TO MENU SCREEN
                                                  E: TERMINATION       S112
```

Figure 2–8. Data Report Form Specifications Screen Display

Figure 2–9 shows entries on this screen for the length of field and the edit code. When indicating field lengths, don't count tab characters. Also, you can't specify the edit code zero (0).



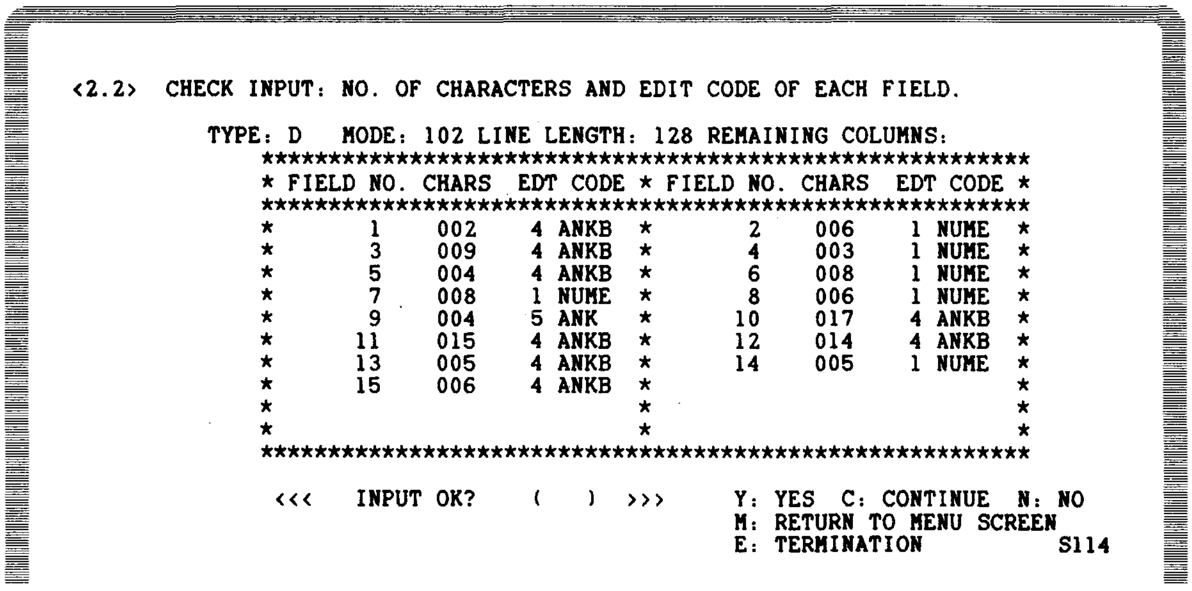
```
<2.1> ENTER NO. OF COLUMNS AND EDIT-CODE FOR EACH FIELD : (YOU MUST ENTER
                          ALL ITEMS IN SEQUENTIAL ORDER BY FIELD NO.)
          TYPE : D   MODE : 102   LINE LENGTH : 128
          ---------------------------------------------------------
          I FIELD NO. CHARS   EDT CODE : FIELD NO. CHARS   EDT CODE I
          ---------------------------------------------------------
          I        1  ( 2    )  ( 4 )   :       2  ( 6    )  ( 1 )   I
          I        3  ( 9    )  ( 4 )   :       4  ( 3    )  ( 1 )   I
          I        5  ( 4    )  ( 4 )   :       6  ( 8    )  ( 1 )   I
          I        7  ( 8    )  ( 1 )   :       8  ( 6    )  ( 1 )   I
          I        9  ( 4    )  ( 5 )   :      10  ( 17   )  ( 4 )   I
          I       11  ( 15   )  ( 4 )   :      12  ( 14   )  ( 4 )   I
          I       13  ( 5    )  ( 4 )   :      14  ( 5    )  ( 1 )   I
          I       15  ( 6    )  ( 4 )   :      16  (      )  (   )   I
          I       17  (      )  (   )   :      18  (      )  (   )   I
          I       19  (      )  (   )   :      20  (      )  (   )   I
          ---------------------------------------------------------
         <<< IS THIS INPUT OK?      (   )  >>>   Y: YES         N: NO
                                                 M: RETURN TO MENU SCREEN
                                                 E: TERMINATION       S112
```

Figure 2–9. Data Report Form Specifications Screen Display with Entries Completed

Once you enter the field items, verify the input, transmit, and enter one of the following options in the last field:

Y	The input is correct. Proceed to the next step of FGEN processing.
N	Mistakes in the input. Display the screen again to correct input. After making changes, press transmit key.
M	Abort the process and redisplay the FGEN function menu screen. Previous input is ignored.
E	Abort the process and display the user logo.

Figure 2–10 shows the results when you don't enter anything in this field or when you enter Y and transmit. This screen confirms the input that was just completed.



```
<2.2>  CHECK INPUT: NO. OF CHARACTERS AND EDIT CODE OF EACH FIELD.

         TYPE: D   MODE: 102 LINE LENGTH: 128 REMAINING COLUMNS:
            *********************************************************
            * FIELD NO. CHARS  EDT CODE * FIELD NO. CHARS  EDT CODE *
            *********************************************************
            *       1     002    4 ANKB  *       2     006    1 NUME  *
            *       3     009    4 ANKB  *       4     003    1 NUME  *
            *       5     004    4 ANKB  *       6     008    1 NUME  *
            *       7     008    1 NUME  *       8     006    1 NUME  *
            *       9     004    5 ANK   *      10     017    4 ANKB  *
            *      11     015    4 ANKB  *      12     014    4 ANKB  *
            *      13     005    4 ANKB  *      14     005    1 NUME  *
            *      15     006    4 ANKB  *                            *
            *                            *                            *
            *                            *                            *
            *********************************************************

             <<<   INPUT OK?    (   )  >>>    Y: YES  C: CONTINUE  N: NO
                                              M: RETURN TO MENU SCREEN
                                              E: TERMINATION         S114
```

Figure 2–10. Data Report Form Specifications Screen Resulting from Blank or Y Entry

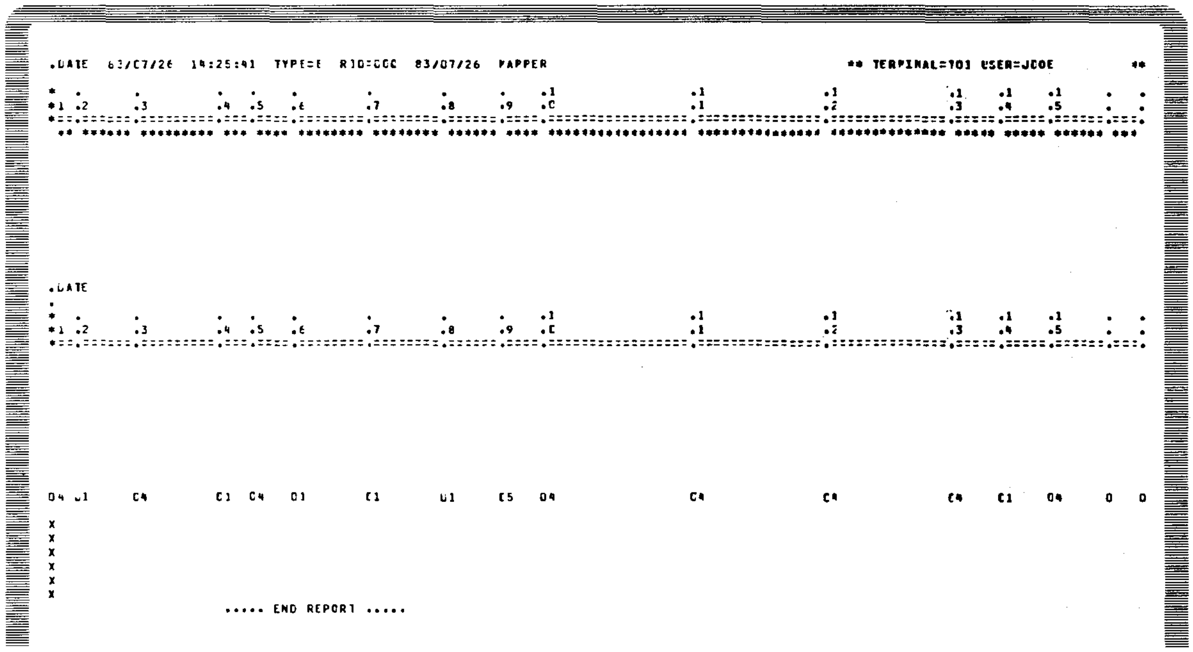
After verifying the contents of the screen, enter N when there are errors and transmit. The screen for requesting item length and edit code reappears, and you can correct the input.

If the contents of the screen are correct, specify Y or C. The MAPPER 80 system handles up to 75 items in one line. However, you can only specify 20 items with one input screen; if there are remaining items, enter C.

C redisplays the screen for requesting item length and edit code. Enter the remaining fields as before.

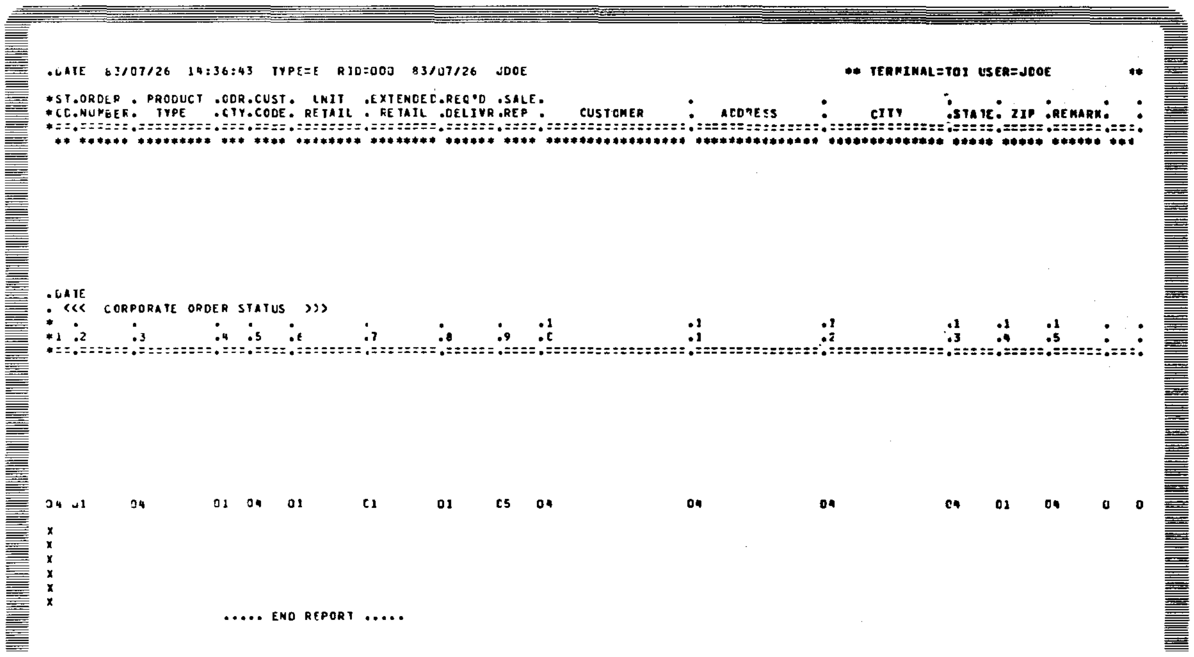
M displays the FGEN function menu screen (Figure 2–7). E displays the user logo screen. Both M or E ignore all previous input.

Y or blank converts RID 0 into the skeleton RID 0 displayed on the screen. (See Figure 2–11.) Figure 2–12 shows RID 0 with field headers and report title.



```
.DATE  83/07/26  14:25:41  TYPE=E  RID=000  83/07/26  MAPPER                        ** TERMINAL=T01 USER=JDOE        **
*  .      .         .   .    .        .        .      .    .1                .1              .1             .1    .1    .1     .   .
*1 .2     .3        .4  .5   .6       .7       .8     .9   .0                .1              .2             .3    .4    .5     .   .
*==.======.=========.===.====.========.========.======.====.=================.===============.==============.=====.=====.======.===.
 ** ****** ********* *** **** ******** ******** ****** **** ***************** *************** ************** ***** ***** ****** ***










.DATE
.
*  .      .         .   .    .        .        .      .    .1                .1              .1             .1    .1    .1     .   .
*1 .2     .3        .4  .5   .6       .7       .8     .9   .0                .1              .2             .3    .4    .5     .   .
*==.======.=========.===.====.========.========.======.====.=================.===============.==============.=====.=====.======.===.










04 01     04        01  04   01       01       01     05   04                04              04             04    01    04     0   0
X
X
X
X
X
X
                    ..... END REPORT .....
```

Figure 2–11. Skeleton RID 0 Screen Display



```
.DATE  83/07/26  14:36:43  TYPE=E  RID=000  83/07/26  JDOE                          ** TERMINAL=T01 USER=JDOE        **
*ST.ORDER . PRODUCT .ODR.CUST.  UNIT  .EXTENDED.REQ'D .SALE.                 .               .              .     .     .      .   .
*CC.NUMBER.  TYPE   .CTY.CODE. RETAIL . RETAIL .DELIVR.REP .    CUSTOMER     .   ADDRESS     .     CITY     .STATE. ZIP .REMARK.   .
*==.======.=========.===.====.========.========.======.====.=================.===============.==============.=====.=====.======.===.
 ** ****** ********* *** **** ******** ******** ****** **** ***************** *************** ************** ***** ***** ****** ***










.DATE
. <<<  CORPORATE ORDER STATUS  >>>
*  .      .         .   .    .        .        .      .    .1                .1              .1             .1    .1    .1     .   .
*1 .2     .3        .4  .5   .6       .7       .8     .9   .0                .1              .2             .3    .4    .5     .   .
*==.======.=========.===.====.========.========.======.====.=================.===============.==============.=====.=====.======.===.










04 01     04        01  04   01       01       01     05   04                04              04             04    01    04     0   0
X
X
X
X
X
X
                    ..... END REPORT .....
```

Figure 2–12. RID 0 Containing Field Header and Report Title

Be careful when there are remaining fields to be specified on the data report form screen. When you leave the INPUT OK option blank and press transmit, all the field definitions are considered complete. If there are remaining fields, enter C and don't leave the option blank.

In Figure 2–11, tab characters are invisible from line 7 to line 16 and from line 22 to line 31; however, tabs are set between fields. These tab characters are counted in the total line length (but not field lengths) indicated at the top of the data report form specification (Figure 2–9).

In addition, in Figure 2–11, the total line length doesn't match the 132-character line length specified in Figure 2–8. Therefore, the four characters are automatically assigned to edit code 4.

When the skeleton RID 0 is displayed, prepare the mask area headers (third and fourth lines) by using the SOE UPDATE manual function. When the length exceeds 80 columns, the screen shifts and the remaining part of the line is completed via the SOE UPDATE function.

When the mask headers are complete, add the report name on line 18 of RID 0 using the SOE UPDATE function.

Then, enter the resume function (RSM) in line 1 or press the F1 function key. The headers from the mask area are automatically copied as headers in the basic report area; with the exception of the modified format definitions, RID 0 is complete.

As soon as the automatic duplication of the header is complete, the screen shown in Figure 2–13 is displayed so that you can complete the definition of the modified formats.

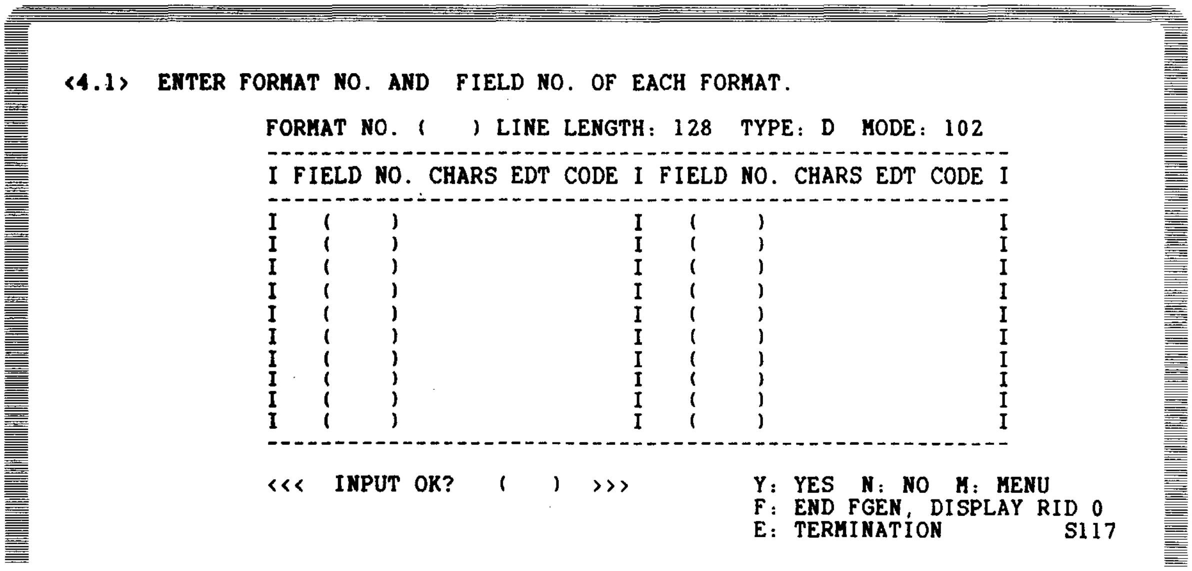
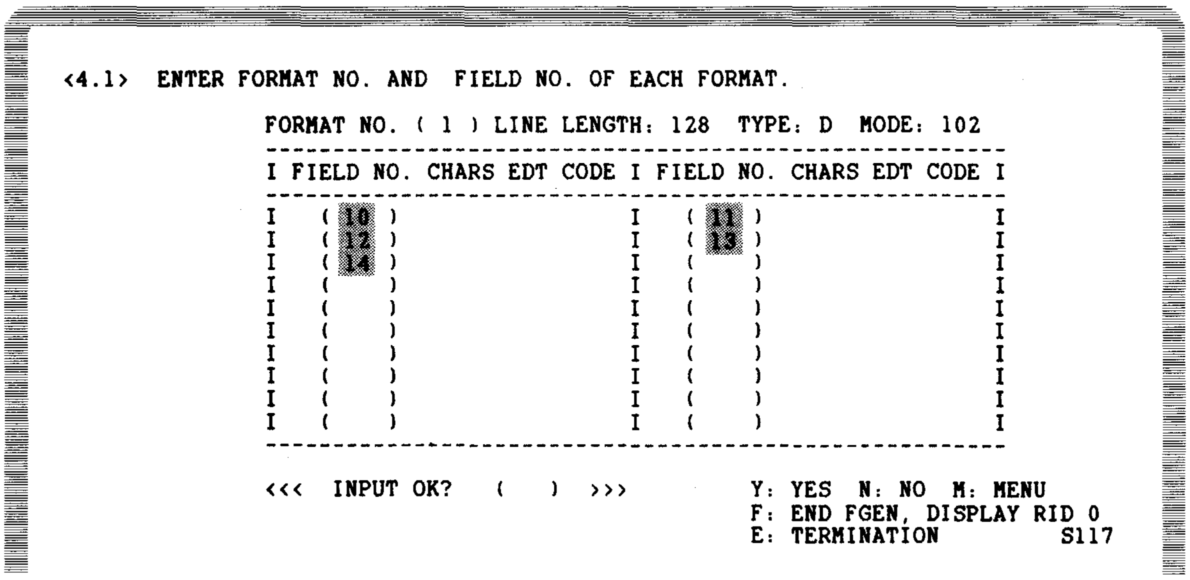

```
<4.1>  ENTER FORMAT NO. AND  FIELD NO. OF EACH FORMAT.

             FORMAT NO. (   ) LINE LENGTH: 128  TYPE: D  MODE: 102
             -----------------------------------------------------
             I FIELD NO. CHARS EDT CODE I FIELD NO. CHARS EDT CODE I
             -----------------------------------------------------
             I   (   )                  I   (   )                  I
             I   (   )                  I   (   )                  I
             I   (   )                  I   (   )                  I
             I   (   )                  I   (   )                  I
             I   (   )                  I   (   )                  I
             I   (   )                  I   (   )                  I
             I   (   )                  I   (   )                  I
             I   (   )                  I   (   )                  I
             I   (   )                  I   (   )                  I
             I   (   )                  I   (   )                  I
             -----------------------------------------------------

             <<<  INPUT OK?   (   )  >>>         Y: YES  N: NO  M: MENU
                                                 F: END FGEN, DISPLAY RID 0
                                                 E: TERMINATION        S117
```

Figure 2–13. Screen Display for Entering Modified Formats

Enter the number of the modified format you want to define, and the numbers of the fields you want to display via this format. (See Figure 2–14.)



```
<4.1>  ENTER FORMAT NO. AND  FIELD NO. OF EACH FORMAT.
                FORMAT NO. ( 1 ) LINE LENGTH: 128  TYPE: D  MODE: 102
                ---------------------------------------------------------
                I FIELD NO. CHARS EDT CODE I FIELD NO. CHARS EDT CODE I
                ---------------------------------------------------------
                I   ( 10 )                 I   ( 11 )                 I
                I   ( 12 )                 I   ( 13 )                 I
                I   ( 14 )                 I   (    )                 I
                I   (    )                 I   (    )                 I
                I   (    )                 I   (    )                 I
                I   (    )                 I   (    )                 I
                I   (    )                 I   (    )                 I
                I   (    )                 I   (    )                 I
                I   (    )                 I   (    )                 I
                I   (    )                 I   (    )                 I
                ---------------------------------------------------------
                <<<  INPUT OK?   (    )  >>>        Y: YES  N: NO  M: MENU
                                                    F: END FGEN, DISPLAY RID 0
                                                    E: TERMINATION          S117
```

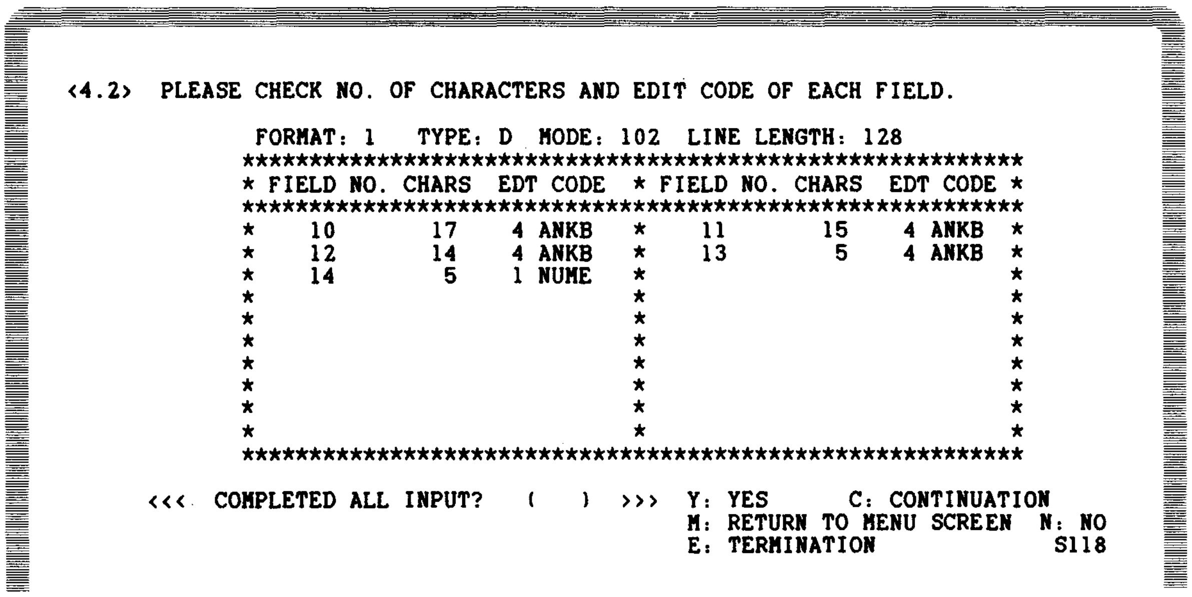
Figure 2–14. Screen Display Showing Modified Format Entries

When you don't define a modified format or when a required definition is complete, enter F, M, or E in the last field and transmit.

F displays the completed RID 0; M calls up the function request menu screen (Figure 2–7); E displays the user logo. In any of these cases, the data you enter is processed.

If the format number and field numbers you entered are correct, enter Y or leave the field blank and transmit.

Figure 2–15 shows the screen display that allows the input to be reconfirmed.



```
<4.2> PLEASE CHECK NO. OF CHARACTERS AND EDIT CODE OF EACH FIELD.

              FORMAT: 1   TYPE: D  MODE: 102  LINE LENGTH: 128
             ***********************************************************
             * FIELD NO. CHARS  EDT CODE  * FIELD NO. CHARS  EDT CODE *
             ***********************************************************
             *    10        17    4 ANKB  *    11        15    4 ANKB *
             *    12        14    4 ANKB  *    13         5    4 ANKB *
             *    14         5    1 NUME  *                           *
             *                            *                           *
             *                            *                           *
             *                            *                           *
             *                            *                           *
             *                            *                           *
             *                            *                           *
             *                            *                           *
             ***********************************************************

      <<< COMPLETED ALL INPUT?  (   )  >>> Y: YES       C: CONTINUATION
                                          M: RETURN TO MENU SCREEN  N: NO
                                          E: TERMINATION            S118
```

Figure 2–15. Data Report Form – Field Confirmation Screen Display

When the contents are incorrect, enter N and the correct data.

When the data is correct, enter C or Y. C indicates continuation when you enter additional items for one type of format. The screen for entering format and field numbers reappears so you can enter the continued items. (See Figure 2–13.)

Y or blank indicates that the definition of one format is complete, and that the modified format specified on RID 0 is registered.

E indicates all necessary formats are defined. Press transmit to complete RID 0.

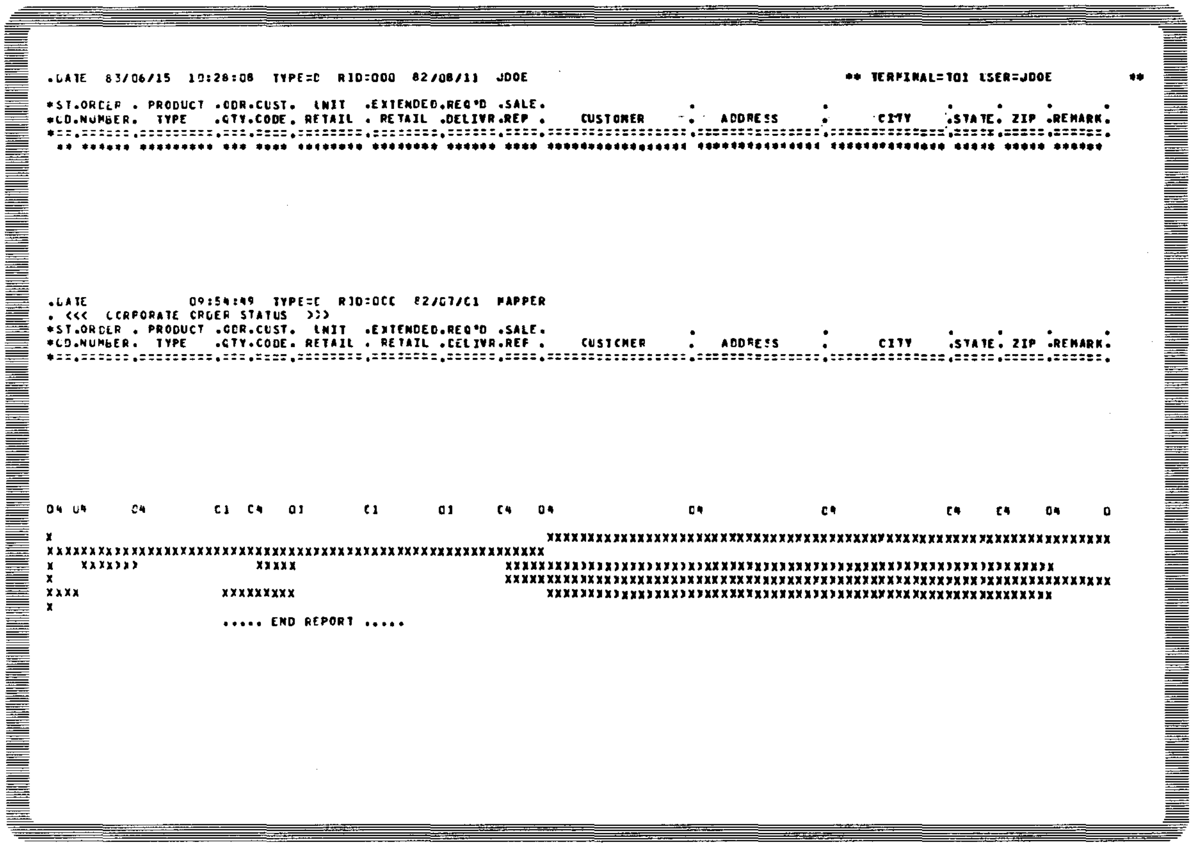
A completed RID 0 terminates the FGEN function.

Confirm the contents of RID 0 on the screen using the shift and roll functions and confirm the result of the modified format. If the confirmation shows irregularities in sections of the skeleton RID 0, correct the problem with the FGEN function, starting at the beginning. If the problem is in the header contents or the modified format section, revise them by using the method outlined in 2.5.

Next, enter the appropriate data in lines 23 to 31 of RID 0 using the SOE UPDATE manual function. You can treat those lines as predefined lines using the ADD LINE manual function. Except for tab lines, set the lines as constants for future reference.

2.4. USING LINE UPDATE FUNCTIONS TO COMPLETE RID 0

In addition to using the FGEN function, you can complete RID 0 by using line update functions. Figure 2–16 shows a sample RID 0 after its completion using line update functions.



```
 1.  .DATE  83/06/15  10:28:08  TYPE=C  RID:000  82/08/11  JDOE                    ** TERMINAL=T01 USER=JDOE       **
 2.
 3.  *ST.ORDER . PRODUCT .ODR.CUST.  UNIT  .EXTENDED.REQ'D .SALE.                  .                  .                 .              .      .      .       .
 4.  *CO.NUMBER.  TYPE   .QTY.CODE. RETAIL . RETAIL .DELIVR.REP .     CUSTOMER     .     ADDRESS      .        CITY     .STATE. ZIP  .REMARK.
 5.  *==.======.=========.===.====.========.========.======.====.================.================.===============.=====.=====.======.
 6.   ** ****** ********* *** **** ******** ******** ****** **** **************** **************** *************** ***** ***** ******
 7.
 8.
 9.
10.
11.
12.
13.
14.
15.
16.
17.  .DATE               09:54:49  TYPE=C  RID:000  82/07/01  MAPPER
18.  . <<<  CORPORATE ORDER STATUS  >>>
19.  *ST.ORDER . PRODUCT .ODR.CUST.  UNIT  .EXTENDED.REQ'D .SALE.                  .                  .                 .              .      .      .       .
20.  *CO.NUMBER.  TYPE   .QTY.CODE. RETAIL . RETAIL .DELIVR.REF .     CUSTOMER     .     ADDRESS      .        CITY     .STATE. ZIP  .REMARK.
21.  *==.======.=========.===.====.========.========.======.====.================.================.===============.=====.=====.======.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32.  D4 U4      C4         C1  C4   O1        C1        O1      C4   O4                   O4                  C4               C4     C4     O4     O
33.
34.  X                                                              XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
35.  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
36.  X   XXXXXXX                XXXXX                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
37.  X                                                         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
38.  XXXX                 XXXXXXXXX                              XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
39.  X
40.                      ..... END REPORT .....
```

Figure 2–16. RID 0 after Completion Using Line Update Functions

To prepare RID 0, use the same method as for updating other MAPPER 80 reports. To complete the RID 0 by using line update functions, you must be able to use the following manual functions: ADD LINE, DELETE LINE, and DUPLICATE LINE. Use these steps to prepare the skeleton RID 0 for a run report:

1. Display the blank RID 0 on the screen.

2. Key in the information from lines 3 to 7, including all periods, equal signs, asterisks, and headings. Leave line 2 blank for future use. Line 7 is a blank tab line. (Tabs must be set manually in the first column and in the columns between fields).

3. Press the transmit key to update lines 3 to 7.

4. When the line length is greater than 81 columns, use the shift function to display columns after column 81.

5. Perform the operations outlined in steps 2 to 4 on the remaining lines. Insert a tab in the last column.

6. Line 7 (an open tab line) is duplicated down to line 16 to finalize the mask region. To do this, enter the following in column 1 of line 7:

   ```
   ▷]10X
   ```

   Operations up to this point complete the mask.

7. The basic report section is produced by duplicating the 16 lines, from line 2 to line 17, beginning at line 18. To do this, key in the following:

   ```
   ▷]1X16
   ```

8. To display the area in the basic report section, key in 17 immediately after the LINE ▷ field on line 0 and transmit.

9. Enter .DATE (from the beginning of the first data line on the screen) on line 17 of RID 0 and transmit.

10. Key in a period in line 18 of RID 0. Key in any comments (e.g., title of the report) and transmit.

11. Because line 6 (corresponding to line 22 of RID 0) remains filled with asterisks, key in the following in column 1 of line 6 to erase this line:

    ```
    ▷]1-
    ```

12. To display the lines below the edit code line, key in 32 immediately after the LINE ▷ field on line 0 and transmit.

13. Enter the edit code in the first data line on the screen (line 32 of RID 0). Tabs are already set in this line as a result of previous operations. Using the tab key advances the cursor, and pressing the ← key reverses the cursor to the preceding column (the tab position). This is where you key in 0 as the edit code. Repeat this operation for all fields until reaching the end of the line.

14. When line length exceeds 81 columns, display the rest of the columns after 81 by using the shift function. Repeat step 13.

15. Next, define the modified formats of RID 0. If the screen was shifted in a previous step, then shift the screen back to column 1 by using the SHIFT function.

16. Define modified formats in the third to eighth data lines on the screen (lines 34–39 of RID 0). Line 34 in RID 0 corresponds to format 1 and line 39 corresponds to format 6. Key Xs directly into all columns of the field and into the tab columns of each format you want to display. Press the transmit key.

17. When line length exceeds 81 columns, repeat step 16 after shifting.

    After defining the formats, use the DISPLAY manual function to display the RID 0 you produced. Key in the format number directly after the FMT ▷ field on line 0. Next, check format status especially when the report is not redisplayed (even if you specified a function on the FMT ▷ field on line 0) because a modified format that was just registered may not be effective.

2.5. EXPANDING AND ALTERING RID 0

There are five ways to expand or alter RID 0:

1. Changing the content of headers

   When you want to change the content of headers only, display RID 0 and use the SOE UPDATE function to change the headers in the mask area. Next, call the FGEN function and request the header copy operation. If you perform this operation, only the headers on RID 0 change. This action doesn't change the headers on the reports that were already generated in this form type.

2. Changing modified formats

   Blanks (nondisplay columns) and Xs (display columns) for each field in each line of the modified formats are defined on RID 0. When you want to change a modified format, modify Xs or blanks in each field by using the SOE UPDATE manual function. When making this change, you must also change the tab columns following each field because they are part of the field. After the change is complete, and you verify that format, display the RID 0 again using the DISPLAY manual function. Then, enter the format number in the FMT ▷ field of line 0 to confirm the result.

3. Changing the report name and predefined lines

   Use the SOE UPDATE manual function to make these changes.

NOTE:

When changing the contents of RID 0, don't use the ADD LINE, DELETE LINE, and DUPLICATE LINE functions because these functions change the lines they refer to and subsequent line numbers. You also risk shifting lines out of position (such as the edit code line and format control line).

4. Changing the edit code

   When changing the field edit code, if it is only a loose check type of change (e.g., edit codes from 1, 2 change to 4), change only the edit code on RID 0, using the UPDATE LINE function.

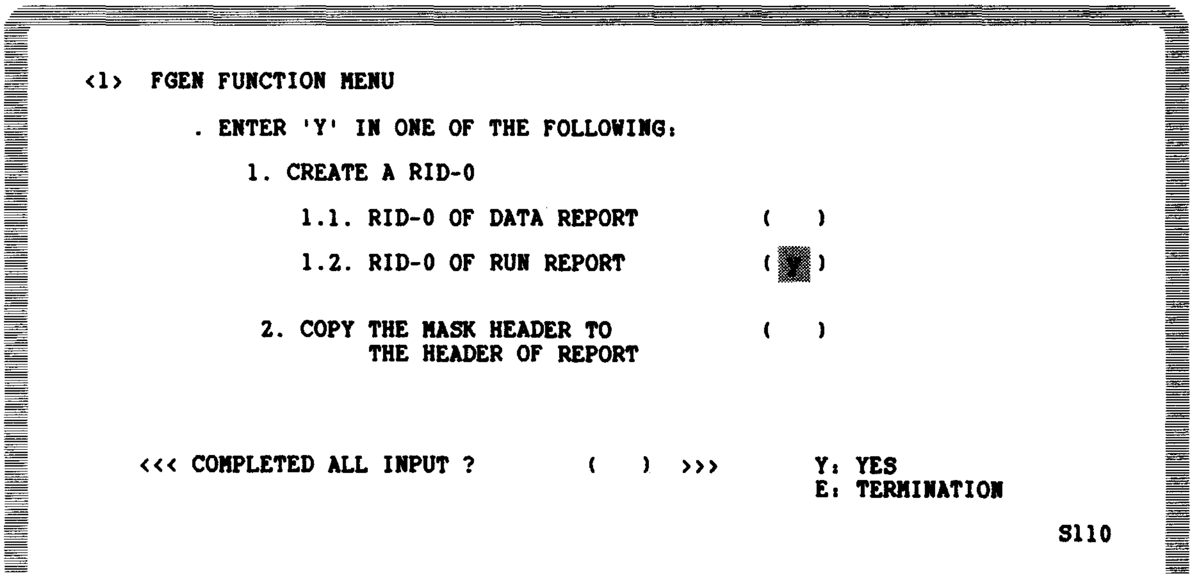
5. Miscellaneous changes

   When you can't make changes by simply changing RID 0, use the following method:

   - Use the MAPLST utility (Section 4) to save all reports in that form type, other than RID 0.
   - Erase all reports except RID 0, using the DELETE REPORT (DR) function.
   - Remodel RID 0 using the FGEN function.
   - Store the reports saved by the MAPLST utility using the MAPLOD utility (Section 3).

2.6. PREPARING RID 0 FOR RUN REPORT

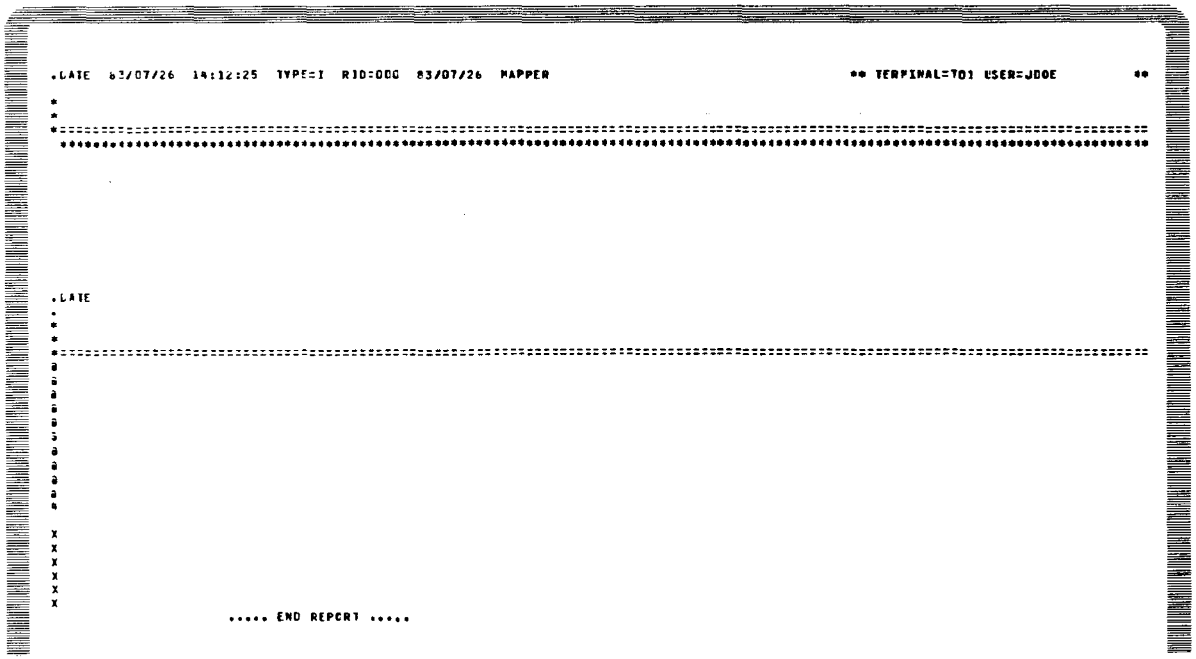
To prepare a run report, call FGEN function (Figure 2–17), designate the form type number, enter the following data to request the type of processing, and transmit.



```
<1>  FGEN FUNCTION MENU

         . ENTER 'Y' IN ONE OF THE FOLLOWING:

            1. CREATE A RID-0

                1.1. RID-0 OF DATA REPORT           (   )

                1.2. RID-0 OF RUN REPORT            ( Y )


             2. COPY THE MASK HEADER TO             (   )
                    THE HEADER OF REPORT



    <<< COMPLETED ALL INPUT ?        (   )  >>>        Y: YES
                                                       E: TERMINATION

                                                                     S110
```

Figure 2–17. FGEN Function Menu Screen Display

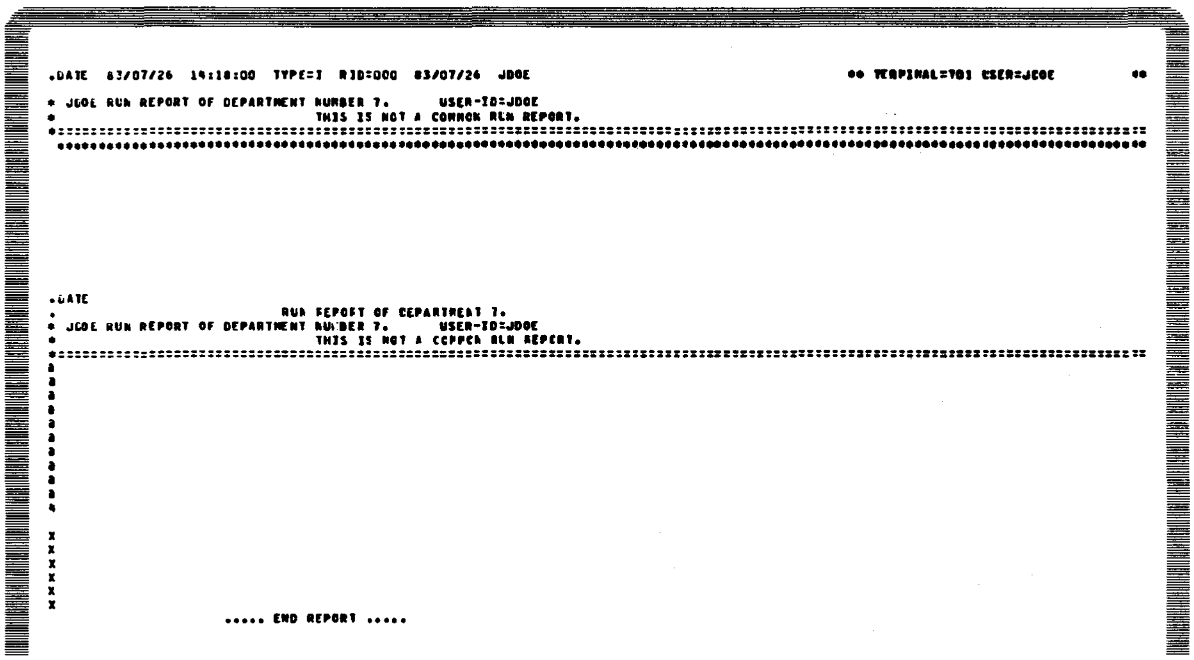
Choosing RID 0 of RUN REPORT on the FGEN menu prepares the skeleton RID 0 for a run report, which is then displayed on the screen. Figure 2–18 shows the skeleton RID 0 for a run report.



```
.DATE  83/07/26  14:12:25  TYPE=I  RID=000  83/07/26  MAPPER                          ** TERMINAL=T01 USER=JDOE         **

*
*
*=====================================================================================================================
 *********************************************************************************************************************










.DATE
.
*
*
*=====================================================================================================================
@
@
@
@
@
@
@
@
@
@
@

X
X
X
X
X
X
                 ..... END REPORT .....
```

Figure 2–18. Skeleton RID 0 for a RUN Report

Using the SOE UPDATE function, enter the headers for the displayed skeleton RID 0 mask area in lines 3, 4, and 18 of the basic report area. Next, enter the resume function (RSM) or the F1 function key to automatically copy the headers in the mask area into the header area of the basic report. The completed RID 0 (Figure 2-19) appears on the screen.



```
 1.  .DATE  83/07/26  14:18:00  TYPE=I  RID=000  83/07/26  JDOE                                       ** TERMINAL=T01 USER=JDOE        **
 2.
 3.  * JDOE RUN REPORT OF DEPARTMENT NUMBER 7.       USER-ID=JDOE
 4.  *                                 THIS IS NOT A COMMON RUN REPORT.
 5.  *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 6.   *****************************************************************************************************************
 7.
 8.
 9.
10.
11.
12.
13.
14.
15.
16.
17.  .DATE
18.  .                              RUN REPORT OF DEPARTMENT 7.
19.  * JDOE RUN REPORT OF DEPARTMENT NUMBER 7.       USER-ID=JDOE
20.  *                                 THIS IS NOT A COMMON RUN REPORT.
21.  *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
22.  a
23.  a
24.  a
25.  a
26.  a
27.  a
28.  a
29.  a
30.  a
31.  a
32.  4
33.
34.  X
35.  X
36.  X
37.  X
38.  X
39.  X
40.                   ..... END REPORT .....
```

Figure 2-19. Completed RID 0 for RUN Report

After this, you establish the predefined lines in the same way as for the normal RID 0.

3. MAPLOD Utility

3.1. WHAT THE MAPLOD UTILITY DOES

MAPLOD is a batch utility that takes data from a spool file or other existing user data and creates new reports, or appends data to existing reports on the MAPPER 80 data base.

By using MAPLOD, large MAPPER 80 reports can easily be constructed from existing system output and master data.

Figure 3–1 shows the control flow for the MAPLOD utility.

You create a spool file from your own program or data utilities output. The print images produced are held in the spool file.

Unused lines (i.e., headers, total lines, etc) of the print images produced for the spool file are removed or converted in the MAPPER 80 report.

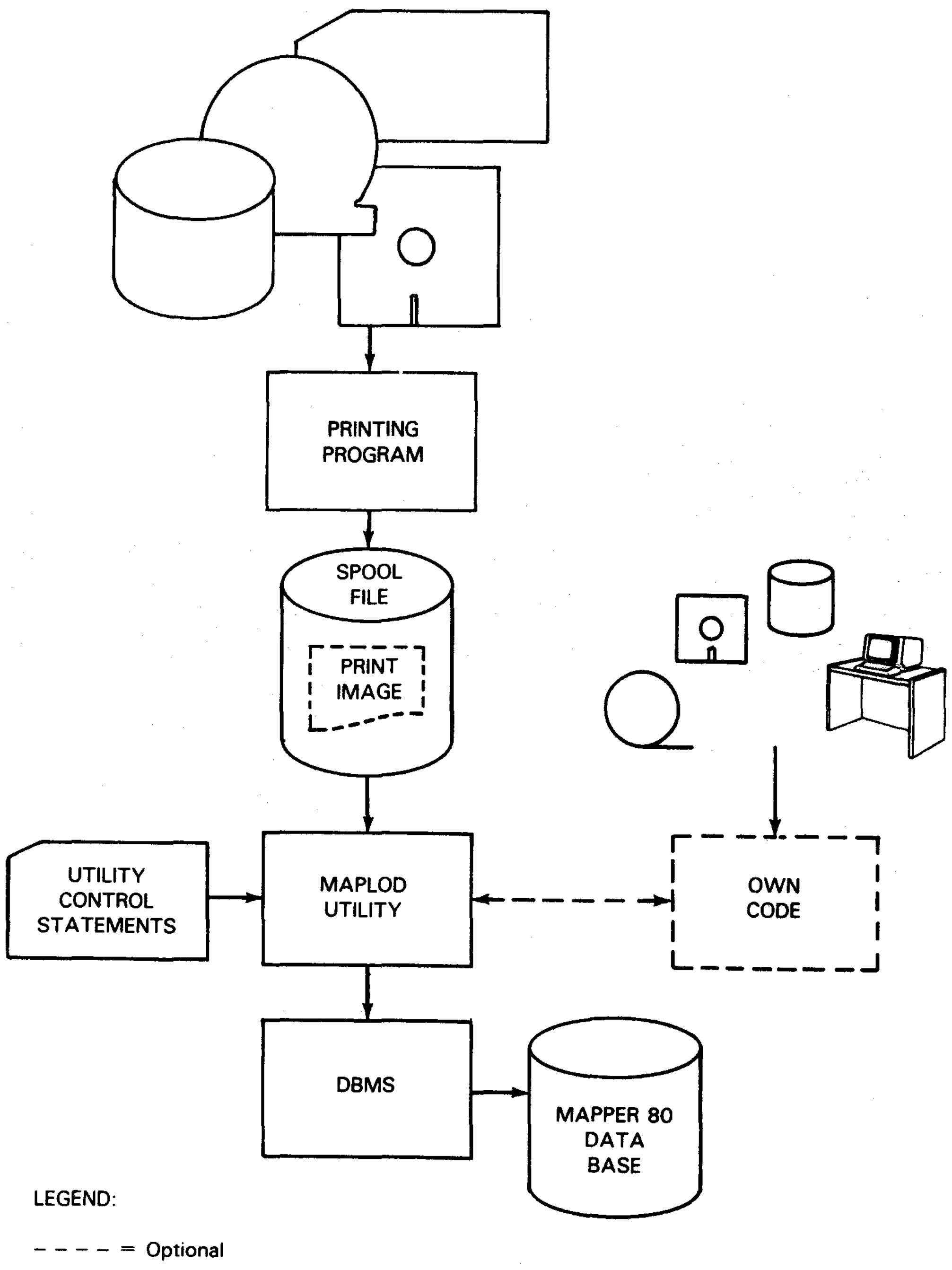
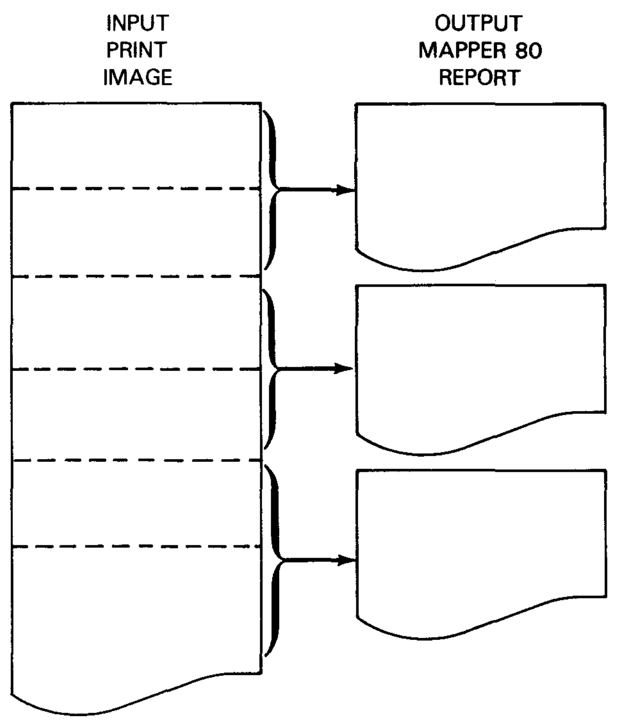


Figure 3–1. Control Flow for MAPLOD Utility Execution

The MAPLOD utility can produce multiple MAPPER 80 reports in one execution.



When converting each line of the print image to a MAPPER 80 report line, you specify the format (alphabetic, numeric, etc) for the items on the line. When the input lines don't conform to the specified format, those lines don't become a part of the MAPPER 80 report. (This is a way to eliminate lines such as headers and totals on the print image.)

Headers on the MAPPER 80 report are copied from header lines of the report type RID 0. Also, tab codes separating the fields are inserted according to RID 0.

MAPLOD can change and output the position and format of each item within the line.

You can create a report tailored to your requirements by writing your own-code routine.

3.2. MAPLOD CONTROL STATEMENTS

MAPLOD control statements specify the attributes of the input print image, your own code, input/output items, and the MAPPER 80 report to be generated. There are four types of control statements:

#INP	Specifies the input print image within the spool file and must be the first control statement.
#OWN	Specifies use of your own code.

#FLD	Defines the position, length, and format of each field on the input print image line.
#OUT	Specifies the mode, type, and RID number of the MAPPER 80 report to be generated.

3.2.1. Identifying Input Form within Spool Print File (#INP)

The #INP control statement is required, and you may specify only one #INP statement. It identifies the print image processed from a possible multiple of print images in the spool file.

Format:

```
#INP FORM={formname}[,JOBN=jobname]
          {STAND1  }
```

where:

FORM=formname
: Is the name of the print form (1 to 8 characters) specified on the SPL or VFB job control statements when the print images on the spool file are created.

FORM=STAND1
: Is the assumed name of the print form when formname was not specified at spool file creation time.

JOBN=jobname
: Is the optional parameter that identifies the print image needed when there are multiple print images on the spool file. The name of the print image alone is not enough to uniquely identify the print image.

For example, when two different jobs create print images that appear in the spool file and neither job assigned a formname to the print image with the SPL or VFB job control statements, both images are assigned the default formname, STAND1. To specify which STAND1 formname is needed, specify the job that created the STAND1 formname.

NOTE:

Use the VFB job control statement to uniquely name each print image. When this is not possible, don't create multiple print images in the same job.

3.2.2. Including Your Own Code in MAPLOD Utility (#OWN)

The #OWN control statement allows you to specify your own-code routine in the MAPLOD utility. Use this statement only once when your own code is included; it must directly follow the #INP statement.

Format:

```
#OWN phasename
```

where:

phasename
: Identifies the phase name provided at link time for your own code, a loadable program that was compiled and linked in a separate job.

3.2.3. Defining Position, Length, and Format of Fields in the Input Form (#FLD)

The #FLD statement specifies the position, length, and data type of fields on the input print image line and the position of the fields on the line of the output report.

Data types of input fields are verified for agreement with the data type specified on the output. When they agree, data is moved to output fields having that field-num in the report line.

When you don't use an output field-num specification, only input verification is performed and the data is not output to a report. This eliminates unwanted lines. You can use multiple #FLD statements or specify parameters for multiple fields within a single statement.

Format:

```
#FLD position,length,format[,field-num]
                    [;position,length,format[,field-num]]...
```

Commas separate individual parameters, and semicolons separate groups of parameters. You can't spread parameters over two #FLD statements.

where:

position
: Specifies the first column of the field in the input line is the first column of the output line. The default column 1 for the output line is the start of the first field on the input line. The maximum column number is 1024 when using your own code.

length
: Defines the length of the input fields (number of bytes).

`format`

Specifies the data type for the input fields. Table 3–1 shows the available data types and their approved contents.

`field-num`

Specifies the relative number of the defined field designated by RID 0 of the report's output form type. Assign it counting from the first fields as field number 1. This parameter is optional and when omitted, verification is performed only on the data format of that input field and that field is not placed on the MAPPER 80 report.

Table 3–1. Data Types and Contents

Edit Code	Approved Contents
1	Numeric fields (0–9). Plus sign (+), minus sign (–), and decimal points are permitted. The field can be blank-filled on the left (i.e., right-justified).
2	Edited numeric fields. Includes attributes of format 1, a trailing minus sign (–), trailing blanks, the dollar sign ($), and commas (e.g., --$2,400.00---).
3	Fields containing only numeric data (0–9); blanks are not permitted.
4	Alphanumeric fields. Blanks, alphabetic, 0–9, and special characters are allowed.
5	Alphanumeric fields where blanks are not permitted.
6	Blank fields for erasing unused lines. Use when scanning certain columns. Must be blank.

The #FLD control statement has two functions:

- To define the conversion of the field between input and output
- To suppress certain lines from the output report by verifying only the data format of the input fields

When converting fields between input and output, the starting position, length, and data format of the fields in the lines of the output report are requested in the edit code of RID 0. Conversion is still performed with certain combinations when the data format in the input doesn't correspond to the output fields. Also, when the length differs, blanks are inserted or truncation occurs according to the output data format (RID 0 edit code). Approved format conversions and editing methods appear in Table 3–2. Refer to Table 2–1 for the edit code of RID 0.

Table 3-2. MAPLOD Utility Field Conversion Rules

Allowable Character Formats		Output Result	Example	When the Length is Different	
Input Format (#FLD)	Output Format (RID 0)			Output Field Longer than Input	Output Field Shorter than Input
1	1	Contents unchanged	−123.45→−123.45	Leading blanks inserted	Truncated on left
1	4	Contents unchanged	+67.89→+67.89	Trailing blanks added	Trun˯ated on right
2	1	Dollar sign ($) and comma are erased. The + or − is moved to the front of the data.	$1,234.5ƀ→ƀƀ1234.5 67,890.1−→ ƀ −67890.1	Leading blanks inserted	Truncated on left
2	2	Sign + or − is moved to the front.	$1234.5−→−$1,234.5	Leading blanks inserted	Truncated on left
2	4	Contents unchanged	$1,234.5−→$1,234.5−	Trailing blanks added	Truncated on right
3	1	Contents unchanged	12345→12345	Leading blanks inserted	Truncated on left
3	4	Contents unchanged	12345→12345	Trailing blanks added	Truncated on right
4	4	Contents unchanged	ABCƀ#1→ABCƀ#1	Trailing blanks added	Truncated on right
5	4	Contents unchanged	XYZ#2→XYZ#2	Trailing blanks added	Truncated on right
5	5	Contents unchanged	XYZ#2→XYZ#2	Trailing blanks added	Truncated on right

3.2.4. Assigning Mode and Type for MAPPER 80 Reports (#OUT)

The #OUT control statement assigns the mode and type for the MAPPER 80 reports being created. Each #OUT control statement produces one output report. You can use multiple statements after the #FLD control statement.

The output field number specified on the #FLD control statement is verified. You select the output field lines, the length of output fields, and data format (edit code) by using the #OUT control statement and the edit code line on RID 0 of the designated type.

Be careful when assigning a type change with multiple #OUT control statements because the output field attributes change.

Format:

```
#OUT user-id,dept-num,[mode-num],[password],type[,[report-num]
        [,first-pg-num,last-pg-num[,E]]]
```

where:

`user-id`
: Assigns a user identification name. When combined with a department number (the same type of verification as the interactive SIGN ON), sets the default mode. Also, your identification (up to 12 characters) is registered in line 1 of the output report.

`dept-num`
: Identifies the department number (up to 3 characters) associated with the user-id.

`mode-num`
: Specifies the number of the mode to which the report is assigned. Mode-num must be an even number. When it is omitted, the report is placed in the default mode number provided with the assigned user-id (0–999).

`password`
: Specifies a password. If the mode you are accessing is the default mode for the user-id, the password is not used. Otherwise, if the mode has a password, you must specify the correct password.

`type`
: Indicates the output report's type (A through I).

`report-num`
Identifies the RID number of the output report. When a report already exists with a RID number from 1 to 999, the existing report is destroyed and the new report is substituted. If the RID number is omitted, the report receives the lowest available number within the designated type.

`first-pg-num`
Specifies the number of the first page of the print image file for input to this report (1 to 9999999).

`last-pg-num`
Specifies the number of the last page of the print image file for input to this report (1 to 9999999). Last-pg-num must have a higher page number than first-pg-num.

NOTES:

1. *When you omit both first and last page values, the whole print image file is used as input.*

2. *When you specify multiple #OUT control statements, you must specify a page number, establish your own-code routine, and specify the last page of the report.*

3. *When using multiple #OUT control statements, assign the page range to avoid overlapping. Some correct and incorrect examples follow:*

   Correct Examples

   ```
   #OUT JDOE,4,,,B,,1,2Ø
   #OUT JDOE,4,,,B,,21,6Ø
   #OUT ACCT,8,,,C,,7Ø,95
   #OUT ACCT,8,,,C,,96,12Ø
   #OUT ACCT,8,,,C,,121
   ```

   Incorrect Examples

   ```
   #OUT PAYR,4,,,B,,1,5Ø
   #OUT ACCT,8,,,C,,3Ø,8Ø
   ```

4. *When you specify letter E, an eject is produced in the output report corresponding to the home skip in the print image file. When printing with the interactive PRINT function or MAPLST utility, the eject line on the MAPPER 80 report occurs after the page changes.*

3.2.5. Appending Data to Existing Reports

If you want to append the new data to an existing report instead of replacing it, include the following job control statement in the MAPLOD execution stream:

```
// SET UPSI,1
```

The new data is appended after the last line of the existing report immediately before the END REPORT line.

3.3. MAPLOD EXECUTION

3.3.1. Execution Environment

Complete the following items before executing the MAPLOD utility:

1. Register the department number and user-id, designated by the #OUT control statement, with the MAPPER 80 software. Refer to the MAPPER 80 operator and coordinator user guide, UP-9737 (current version).

2. Complete the form generation for the report type specified in the MAPLOD execution stream. (RID 0 must be completed. See Section 2.)

3. Start the DBMS. Refer to the MAPPER 80 operator and coordinator user guide, UP-9737 (current version).

4. Prepare the input spool file.

When MAPLOD execution is completed, the status (normal end or error end) displays on the system console. On an error end status, bit 1 (4016) of the UPSI byte is set.

3.3.2. Preparing the Input Spool File

Before executing the MAPLOD utility, you must store a print image in the spool file and hold the spool file by using one of the following procedures:

1. Include either of these statements in the job control stream that creates the print image:

   ```
   // SPL HOLD
   ```

   or

   ```
   // SPL RETAIN
   ```

2. The operator can use either of these commands:

   ```
   HOLD SPL
   ```

   or

   ```
   PR HOLD
   ```
   (issued to the output writer)

For further details, refer to the spooling and job accounting concepts and facilities, UP-8869 (current version).

Here is an example that shows how to prepare a print image:

```
 1      // JOB PRTOUT,,9000
 2      // DVC 20
 3      // SPL HOLD,,,,,FORMA
 4      // LFD PRNTR
 5      // DVC 50  // VOL MAPPER
 6      // LBL MASTER1  // LFD INPUT1
 7      // EXEC DATA
 8      // PARAM CONTROL=NO
 9      // PARAM DISPLAY=L
10      /$
11       UDP B=(132,132),MKR=(0)
12      /*
13      /&
```

The // SPL HOLD statement (line 3) with the form name FORMA specifies holding the data utility output print image on the spool file.

The next device assignment set beginning in line 5 assigns the file for input.

The parameters (lines 8 and 9) after the data utilities execution statement inhibit the printing of data utilities control statements and termination information. This reduces the excess print image as MAPLOD input. If you can eliminate this information by the MAPLOD #FLD control statement, it can be printed.

Line 11 indicates that the MIRAM file should be printed sequentially.

Figure 3-2 shows a sample spool file print image generated by the PRTOUT job stream.

```
  1      ST 820602 JH BEST 732732 01121 SUNGLASS1 830602 830929 830930 J2060
  2      AL 811002 SG HOPP 732733 01122 EYEGLASS9 821222 830121 830122 J2061
  3      ST 820704 KC BEST 732734 01123 EYEGLASS7 811111 811211 811212 J2080
  4      EZ 820801 RR HARK 738756 02212 SUNGLASS5 831214 831215 831217 J2084
  5      AL 820404 RX BACH 742312 02225 SUNGLASS3 820406 820410 820430 J2000
  6      ST 820602 JH BEST 732732 01121 SUNGLASS1 830602 830929 830930 J2060
  7      AL 811002 SG QUIP 732733 01122 EYEGLASS7 821222 830121 830122 J2061
  8      ST 820704 KC QUIP 732734 01123 EYEGLASS9 811111 811211 811212 T2080
  9      EZ 820801 RR HOPP 738756 02212 SUNGLASS5 831214 831215 831217 T2084
 10      ST 820602 JH QUIP 732732 01121 SUNGLASS1 830602 830929 830930 T2060
 11      AL 811002 SG SITE 732733 01122 EYEGLASS7 821222 830121 830122 T2061
 12      ST 820704 KC QUIP 732734 01123 EYEGLASS0 811212 811211 811212 T2080
 13      AL 811002 SG EYES 732733 01122 SUNGLASS7 821222 830121 830122 T2061
 14      CC 820704 KC BEST 732734 01123 EYEGLASS9 811212 811211 811212 T2080
 15      EZ 820801 RR HARK 738756 02212 SUNGLASS5 831214 831215 831217 T2084
 ≈                                                                      ≈
 71      CC 820704 KC BEST 732734 01123 EYEGLASS9 811212 811211 811212 T2080
 72      ST 820704 KC BEST 732734 01123 EYEGLASS7 811111 811211 811212 J2080
 73      EZ 820801 RR HARK 738756 02212 SUNGLASS5 831214 831215 831217 J2084
 74      AL 820404 RX BACH 742312 02225 SUNGLASS3 820406 820410 820430 J2000
 75      ST 820602 JH BEST 732732 01121 SUNGLASS1 830602 830929 830930 J2060


         1234567891111111111222222222233333333334444444444555555555566666666
                  0123456789012345678901234567890123456789012345678901234567
```

Figure 3-2. Sample Spool File Print Image

Each of the example MAPLOD executions in this section use this spool file.

NOTE:

This spool file was generated from a MIRAM file, and its print image was created by using a data utilities job stream. This is not the only way to create a print image for MAPLOD input. You can use any existing application program that produces printed output.

3.3.3. MAPLOD Run Stream Examples

Once the input spool file is prepared, you are ready to execute the MAPLOD utility by using the following run stream:

```
JOB jobname
// DVC 2Ø   // LFD PRNTR
                    (include other device assignment sets here)
// EXEC MAPLOD
/$
                    (MAPLOD control statements)
/*
/&
// FIN
```

The following are examples of MAPLOD job streams, including utility control statements. These streams use the sample spool file (Figure 3–2) generated by the PRTOUT job stream.

NOTE:

RID numbers in the following MAPLOD job streams must not coincide with existing MAPPER 80 reports on the JDOE training data base (mode 102). To avoid overwriting the existing report with the same mode, type, and RID, you may set the UPSI byte to 1 (see 3.2.5) or change the RID number in the execution stream.

The first sample stream produces a single output report. There is no own-code routine.

```
 1       // JOB TESTLOD1
 2       // DVC 20   // LFD PRNTR
 3       // EXEC MAPLOD
 4       /$
 5       #INP FORM=FORMA,JOBN=PRTOUT
 6       #FLD 1,2,4,1;4,6,1,2;11,2,4,3
 7       #FLD 14,4,4,8;19,6,1,5;26,5,1,7
 8       #FLD 32,9,4,4;42,6,1,9;49,6,1,10
 9       #FLD 56,6,1,11;63,5,4,12
10       #OUT JDOE,7,102,,B,40
11       /*
12       /&
```

The first DVC control statement (line 2) allocates the printer.

The #INP statement (line 5) names the print image FORMA on the spool file as input.

The #FLD statements (lines 6–9) move the fields on the print image into the output report fields, while verifying the data format:

- The first #FLD statement moves 2 columns of data (beginning with column 1) to the first output field; 6 columns of data (beginning with column 4) to the second output field; and 2 columns of data (beginning with column 11) to the third output field, etc.
- The second #FLD statement moves 4 columns of data (beginning with column 14) to output field 8; 6 columns of data (beginning with column 19) to the fifth output field; and 5 columns of data (from column 26) to output field 7.
- The third #FLD statement moves 9 columns of data (beginning with column 32) to output field 4; 6 columns of data (beginning with column 42) to output field 9; and 6 columns of data (from column 49) to the tenth output field.

- The last #FLD statement moves 6 columns of data (from column 56) to output field 11, and 5 columns of data (from column 63) to the twelfth output field.

  NOTE:

  These #FLD statements are the same on each of the MAPLOD execution examples, making it possible for each example to use the same spool file print image.

- The #OUT statement indicates creation of a report with department number 7, user-id JDOE, with RID 40, type B, and mode 102.

Figure 3-3 shows a MAPPER 80 display of the report resulting from the first sample MAPLOD execution (TESTLOD1).

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD>           >
.DATE  83/10/11  12:51:19  TYPE=B  RID=040  83/10/10  JDOE          <   81 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 ST 820602 JH SUNGLASS1 732732        01121 BEST 830602 830929 830930 J2060
 AL 811002 SG EYEGLASS9 732733        01122 HOPP 821222 830121 830122 J2061
 ST 820704 KC EYEGLASS7 732734        01123 BEST 811111 811211 811212 J2080
 EZ 820801 RR SUNGLASS5 738756        02212 HARK 831214 831215 831217 J2084
 AL 820404 RX SUNGLASS3 742312        02225 BACH 820406 820410 820430 J2000
 ST 820602 JH SUNGLASS1 732732        01121 BEST 830602 830929 830930 J2060
 AL 811002 SG EYEGLASS7 732733        01122 QUIP 821222 830121 830122 J2061
 ST 820704 KC EYEGLASS9 732734        01123 QUIP 811111 811211 811212 T2080
 EZ 820801 RR SUNGLASS5 738756        02212 HOPP 831214 831215 831217 T2084
 ST 820602 JH SUNGLASS1 732732        01121 QUIP 830602 830929 830930 T2060
 AL 811002 SG EYEGLASS7 732733        01122 SITE 821222 830121 830122 T2061
 ST 820704 KC EYEGLASS0 732734        01123 QUIP 811212 811211 811212 T2080
 AL 811002 SG SUNGLASS7 732733        01122 EYES 821222 830121 830122 T2061
 CC 820704 KC EYEGLASS9 732734        01123 BEST 811212 811211 811212 T2080
 EZ 820801 RR SUNGLASS5 738756        02212 HARK 831214 831215 831217 T2084

 CC 820704 KC EYEGLASS9 732734        01123 BEST 811212 811211 811212 T2080
 ST 820704 KC EYEGLASS7 732734        01123 BEST 811111 811211 811212 J2080
 EZ 820801 RR SUNGLASS5 738756        02212 HARK 831214 831215 831217 J2084
 AL 820404 RX SUNGLASS3 742312        02225 BACH 820406 820410 820430 J2000
 ST 820602 JH SUNGLASS1 732732        01121 BEST 830602 830929 830930 J2060
                 ..... END REPORT .....
```

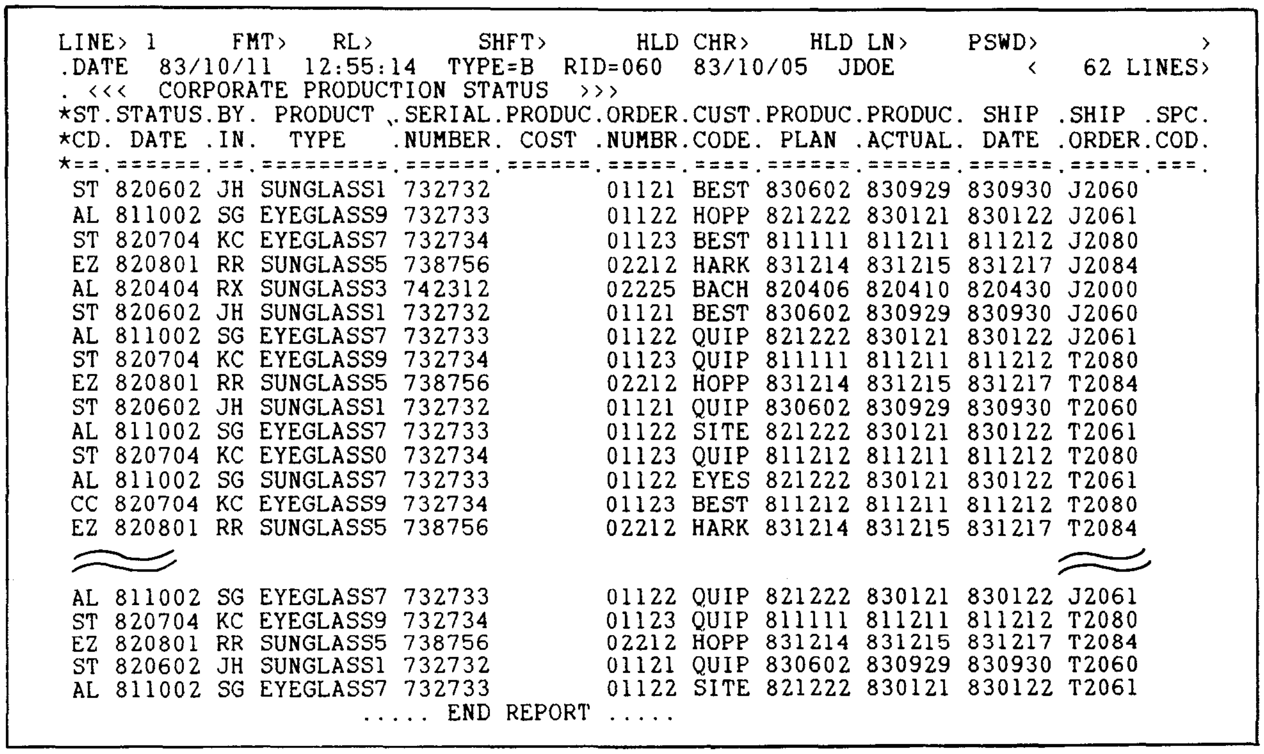
Figure 3-3. Display of RID 40B Created by TESTLOD1

The next example produces multiple output reports. There is no own-code routine.

```
1       // JOB TESTLOD2
2       // DVC 20   // LFD PRNTR
3       // EXEC MAPLOD
4       /$
5       #INP FORM=FORMA,JOBN=PRTOUT
6       #FLD 1,2,4,1;4,6,1,2;11,2,4,3
7       #FLD 14,4,4,8;19,6,1,5;26,5,1,7
8       #FLD 32,9,4,4;42,6,1,9;49,6,1,10
9       #FLD 56,6,1,11;63,5,4,12
10      #OUT JDOE,7,102,,B,60,1,2
11      #OUT JDOE,7,102,,B,61,3,4
12      /*
13      /&
```

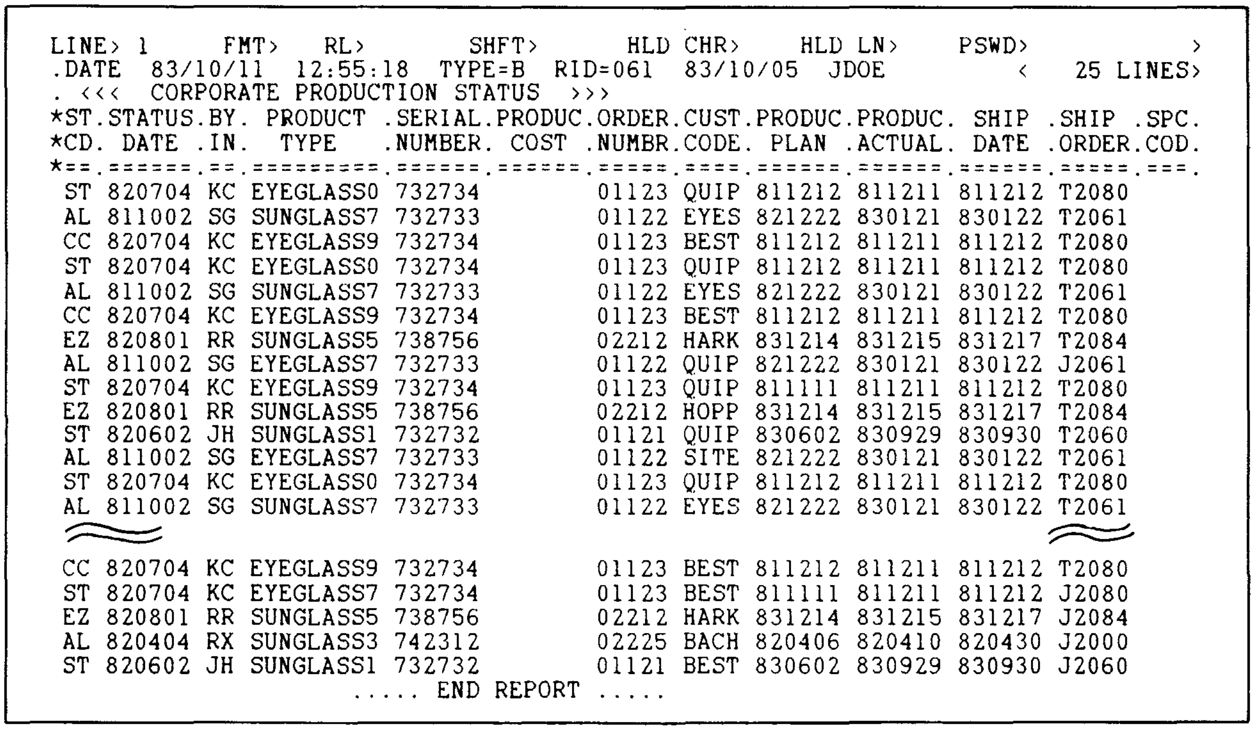
This job stream (TESTLOD2) is the same as the first example (TESTLOD1) except for an additional #OUT control statement. The two #OUT control statements refer to different RID numbers. The #OUT control statement in line 10 produces a MAPPER 80 report in RID 60 using the print image on pages 1 and 2 of the spool file. The #OUT control statement on line 11 uses spool file pages 3 and 4 of the same print image to create RID 61.

Figures 3–4 and 3–5 show MAPPER 80 displays resulting from the second MAPLOD execution (TESTLOD2). TESTLOD2 created RIDs 60B and 61B in mode 102.



```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>     PSWD>          >
.DATE  83/10/11  12:55:14  TYPE=B  RID=060  83/10/05  JDOE           <   62 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT  .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE    .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 ST 820602 JH SUNGLASS1 732732        01121 BEST 830602 830929 830930 J2060
 AL 811002 SG EYEGLASS9 732733        01122 HOPP 821222 830121 830122 J2061
 ST 820704 KC EYEGLASS7 732734        01123 BEST 811111 811211 811212 J2080
 EZ 820801 RR SUNGLASS5 738756        02212 HARK 831214 831215 831217 J2084
 AL 820404 RX SUNGLASS3 742312        02225 BACH 820406 820410 820430 J2000
 ST 820602 JH SUNGLASS1 732732        01121 BEST 830602 830929 830930 J2060
 AL 811002 SG EYEGLASS7 732733        01122 QUIP 821222 830121 830122 J2061
 ST 820704 KC EYEGLASS9 732734        01123 QUIP 811111 811211 811212 T2080
 EZ 820801 RR SUNGLASS5 738756        02212 HOPP 831214 831215 831217 T2084
 ST 820602 JH SUNGLASS1 732732        01121 QUIP 830602 830929 830930 T2060
 AL 811002 SG EYEGLASS7 732733        01122 SITE 821222 830121 830122 T2061
 ST 820704 KC EYEGLASS0 732734        01123 QUIP 811212 811211 811212 T2080
 AL 811002 SG SUNGLASS7 732733        01122 EYES 821222 830121 830122 T2061
 CC 820704 KC EYEGLASS9 732734        01123 BEST 811212 811211 811212 T2080
 EZ 820801 RR SUNGLASS5 738756        02212 HARK 831214 831215 831217 T2084

 AL 811002 SG EYEGLASS7 732733        01122 QUIP 821222 830121 830122 J2061
 ST 820704 KC EYEGLASS9 732734        01123 QUIP 811111 811211 811212 T2080
 EZ 820801 RR SUNGLASS5 738756        02212 HOPP 831214 831215 831217 T2084
 ST 820602 JH SUNGLASS1 732732        01121 QUIP 830602 830929 830930 T2060
 AL 811002 SG EYEGLASS7 732733        01122 SITE 821222 830121 830122 T2061
                  ..... END REPORT .....
```

Figure 3–4. Display of RID 60B Created by TESTLOD2



```
LINE> 1     FMT>   RL>       SHFT>       HLD CHR>    HLD LN>     PSWD>            >
.DATE  83/10/11  12:55:18  TYPE=B  RID=061  83/10/05  JDOE          <   25 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 ST 820704 KC EYEGLASS0 732734        01123 QUIP 811212 811211 811212 T2080
 AL 811002 SG SUNGLASS7 732733        01122 EYES 821222 830121 830122 T2061
 CC 820704 KC EYEGLASS9 732734        01123 BEST 811212 811211 811212 T2080
 ST 820704 KC EYEGLASS0 732734        01123 QUIP 811212 811211 811212 T2080
 AL 811002 SG SUNGLASS7 732733        01122 EYES 821222 830121 830122 T2061
 CC 820704 KC EYEGLASS9 732734        01123 BEST 811212 811211 811212 T2080
 EZ 820801 RR SUNGLASS5 738756        02212 HARK 831214 831215 831217 T2084
 AL 811002 SG EYEGLASS7 732733        01122 QUIP 821222 830121 830122 J2061
 ST 820704 KC EYEGLASS9 732734        01123 QUIP 811111 811211 811212 T2080
 EZ 820801 RR SUNGLASS5 738756        02212 HOPP 831214 831215 831217 T2084
 ST 820602 JH SUNGLASS1 732732        01121 QUIP 830602 830929 830930 T2060
 AL 811002 SG EYEGLASS7 732733        01122 SITE 821222 830121 830122 T2061
 ST 820704 KC EYEGLASS0 732734        01123 QUIP 811212 811211 811212 T2080
 AL 811002 SG SUNGLASS7 732733        01122 EYES 821222 830121 830122 T2061

 CC 820704 KC EYEGLASS9 732734        01123 BEST 811212 811211 811212 T2080
 ST 820704 KC EYEGLASS7 732734        01123 BEST 811111 811211 811212 J2080
 EZ 820801 RR SUNGLASS5 738756        02212 HARK 831214 831215 831217 J2084
 AL 820404 RX SUNGLASS3 742312        02225 BACH 820406 820410 820430 J2000
 ST 820602 JH SUNGLASS1 732732        01121 BEST 830602 830929 830930 J2060
                     ..... END REPORT .....
```

Figure 3–5. Display of RID 61B Created by TESTLOD2

3.4. YOUR OWN CODE

3.4.1. What Your Own Code Contains

Control transfers to your own code when the print image on the spool file is read in. You should be careful that no items are transferred when the end of the print image is reached.

The format of transferred data can be a maximum of 171 bytes. Table 3–3 defines the position, length, and contents of data transferred to an output report.

Table 3-3. Description of Data Transferred to Output Reports

Byte Position Number	Number of Bytes	Contents
1 to 3	3	Mode number for intended report output
4	1	Type of intended output report
5 to 7	3	RID number of report
8	1	Indicates print image processing; the zero is transferred to your own code and is used as an indicator to MAPLOD.
9 to 10	2	The length of the print image is set in binary (0–160).
11	1	Line control code; set as the control code for hexadecimal form positioning instructions: ■ When print image length is 0: X'BF' means skip to top of form. Other than X'BF' means advance a line or skip to fixed line. ■ When print image length is not 0: X'B9' means skip to top of form and print. Other than X'B9' means advance a line or skip to fixed line and print.
12 to 171	Maximum 160	Print image

The following items describe how to process a spool file, add data from a MIRAM file, and exit your own code:

- Four indicators (0–3) tell MAPLOD how to handle each line from the spool file. Set these indicators in your own code.

  MAPLOD passes one line at a time from the spool file to your own code. Your own code then sets the indicator to tell MAPLOD how to handle that line.

  When your own code reads from a MIRAM file, that record becomes the line to be processed by MAPLOD. You don't lose the spool file record; however, MAPLOD processes it on the next call to your own code when the MIRAM file is not read.

MAPLOD calls your own code once for each line to be processed. When there are 50 lines in the spool file and 25 records in the MIRAM file, MAPLOD calls your own code at least 50 times and once more for every record read from the MIRAM file. Your own code sets the indicator each time own code is called. Indicator settings and their meanings follow.

0 Indicates that MAPLOD verifies the input line from the spool file or the MIRAM file and sends the input line to the MAPPER 80 report specified on the #OUT control statement. At verification, MAPLOD checks the data format or each field to assure that they match the RID 0 of the MAPPER 80 report. When there is a mismatch, MAPLOD doesn't place the line in the MAPPER 80 report. After MAPLOD processes that line, it reads the next line and passes control to your own code.

1 Indicates that MAPLOD verifies and sends the line to the MAPPER 80 report. However, MAPLOD doesn't read the next spool file record. Instead, MAPLOD passes control to your own code.

2 Indicates that MAPLOD ignores processing the current line. It reads the next line from the spool file and returns control to your own-code routine. Use this indicator to skip lines on the spool file.

3 Tells MAPLOD to go to the next #OUT control statement. This new #OUT control statement changes the mode, type, or RID number. When MAPLOD completes processing the current report, it begins to send the new report specified on the new #OUT control statement. Control then passes to your own code.

  If a new #OUT control statement doesn't exist, MAPLOD stops processing the current report and doesn't return control to your own code. MAPLOD execution terminates after placing an *** END REPORT *** line in the MAPPER 80 report.

- Change the length of line and the line control code via your own code.

- Your own code can add data to the report from internal calculation, a MIRAM file, etc. MAPLOD treats that data as if it were read from the spool file.

- You can expand the line from the spool file to a size of 1024 bytes and return to MAPLOD. Because the line returned to MAPLOD can be 1024 bytes and can be sent out as a MAPPER 80 report, you must watch the line.

- MAPLOD can process the MIRAM file without any input from the spool file when you use indicator 1 and read from a MIRAM file; however, a spool file must always exist even though your own code may not use it.

- Don't change the mode number, type, and RID number in your own code. The #OUT control statement contains this information, and your own code should conform.

3.4.2. How to Use Your Own Code

Your own code handles several types of processing:

- Adding fields not in the print image to the report

  Defined constants, calculation results, etc, throughout the print image reference information from your file (such as names). You can add other similar information before control returns to MAPLOD. In this case, use processing indicator 0.

- Eliminating unnecessary lines

  Use your own code when data format verification was performed only by a #FLD control statement and when a large number of lines were not cleared. Unnecessary output lines are selectively cleared and control returns to MAPLOD with processing indicator 2.

- Combining multiple lines into one line in the report

  After making one report line, control returns to MAPLOD via the processing indicator 2. After the print image changes to one line, control returns to MAPLOD with the 0 indicator.

- Choosing a page for multiple report production in one execution

  When you specify multiple #OUT control statements, your own code should decide in advance the print image page number for an #OUT control statement and whether to switch reports in the middle of a page. You can use processing indicator 3.

- Producing a MAPPER 80 report directly from a user file

  Within your own code, the file is read and handled as the print image. Indicator 1 returns control to MAPLOD. When the file is completely processed, indicator 3 may be used to return to MAPLOD.

You must change packed format and decimal data in your file to display format for future handling. However, when control passes to MAPLOD with the user file records handled as a print image, actual editing of the output report is unnecessary. A maximum of 1024 bytes of data are transferred to MAPLOD and you can assign up to 1024 input field positions with #FLD control statements. Thus, in your own code, only the display format is converted. To choose the necessary output fields and designate their order on the report, use the #FLD control statement.

With this type of processing, it is necessary to reserve a minimum of a 1-line print image on the spool file. (The content of the print image is not important.)

In addition, because the DMS file in MAPLOD (MAPPER 80 data base) is being used, you must be careful that your own file doesn't use the DMS file.

3.4.3. Executing Your Own Code

Prepare your own code for the MAPLOD utility as a called program and place it in the same load library where MAPLOD resides. Your own-code module is linked to MAPLOD at execution time through the OS/3 dynamic load function. Whenever you use own code with MAPLOD, you must specify the job main storage size. The exact size needed is the sum of MAPLOD plus your linked own code size.

3.4.4. Example of Your Own Code

The following explanation describes two applications of user own code:

- Job stream TESTLOD3 creates multiple reports whose data lines are determined by the ZOWNCD own-code routine (Figure 3–6).
- Job stream TESTLOD4 also creates multiple reports using the same print images controlled by a different own-code routine (YOWNCD) (Figure 3–8). However, in addition to the spool file, this job uses a MIRAM file as input to MAPLOD.

The TESTLOD3 job sends each record of the print image to own code ZOWNCD (Figure 3–6). By setting an indicator (0–3), this own code determines how MAPLOD processes each line of the print image.

```
// JOB TESTLOD3,,18000
// DVC 20   // LFD PRNTR
// DVC 200  // UID $Y$MAS  // LFD WORKST
// EXEC MAPLOD
/$
#INP FORM=FORMA,JOBN=PRTOUT
#OWN ZOWNCD
#FLD 1,2,4,1;4,6,1,2;11,2,4,3
#FLD 14,4,4,8;19,6,1,5;26,5,1,7
#FLD 32,9,4,4;42,6,1,9;49,6,1,10
#FLD 56,6,1,11;63,5,4,12
#OUT JDOE,7,102,,B,80
#OUT JDOE,7,102,,B,82
/*
/&
```

NOTE:

The TESTLOD3 job stream assigns a workstation because the own code requires it. Use TESTLOD3 job stream to execute the first sample own-code routine shown in Figure 3–6. In this MAPLOD user own-code run, the user determines the resulting data in RIDs 80B and 82B by the indicator he enters at the workstation. (See Figure 3–6, line 56.)

The ZOWNCD own-code routine shown in Figure 3–6 uses a workstation to display information so that you can see the print image being processed. This example also clarifies how to use MAPLOD indicators in your own code.

To do this, the own code displays a prompt message to accept the indicator from the workstation (in this example) before each record is processed. The indicator tells MAPLOD how to process each record (see 3.4.1). There are other ways of setting indicators in your own code (e.g., an IF statement).

```
// JOB MPLDOWN1
// DVC 20  // LFD PRNTR
// WORK1
// WORK2
// WORK3
// EXEC COBL74
/$
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  MYOWN.
      *REMARKS.     THIS PROGRAM ACTS AS A SUBROUTINE TO
      *             A MAPLOD JOB USED TO CREATE A MAPPER
      *             REPORT FROM A SPOOLFILE.
      * * * * * * * * * * * * *  * * * * * * * * * * * *
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. UNIVAC-OS3.
       OBJECT-COMPUTER. UNIVAC-OS3.
       SPECIAL-NAMES.
           SYSWORK IS WRKSTN ASSIGN TO WORKST.

      *
      * * * * * * * * * * * * *  * * * * * * * * * * * *
      *
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  PROGRAM-INDICATOR.
           05  REC                         PIC 9(3) VALUE ZEROS.
           05  LIVE-INDIC                  PIC 9 VALUE ZERO.
       LINKAGE SECTION.
       01  OWN-DATA.
           02  MODE-NO                  PIC 9(3).
           02  TYPEX                    PIC X.
           02  RID-NO                   PIC 9(3).
           02  INDICATOR                PIC 9.
           02  IMAGE-LENGTH             PIC 9(4)  COMP.
           02  CONTROL-CODE             PIC X.
           02  PRINT-IMAGE              PIC X(90).
      *
      * * * * * * * * * * * * *  * * * * * * * * * * * *
      *
       PROCEDURE DIVISION       USING OWN-DATA.
      *
      * * * * * * * * * * * * *  * * * * * * * * * * * *
      *
       MAIN-LINE.
           DISPLAY '* * * * * * * *  * * *  * * * *' UPON WRKSTN.
           DISPLAY 'E N T E R I N G  O W N  C O D E' UPON WRKSTN.
           MOVE 102 TO MODE-NO.
           MOVE 'B' TO TYPEX.
           MOVE 90 TO IMAGE-LENGTH.
           COMPUTE REC = REC + 1
           DISPLAY 'PRINT IMAGE NUMBER ' , REC , ' IS:' UPON WRKSTN.
           DISPLAY PRINT-IMAGE UPON WRKSTN.
```

Figure 3–6. ZOWNCD Own-Code Routine Using Spool File Input and MAPLOD Indicators (Part 1 of 2)

```
55                 DISPLAY 'ENTER PROCESSING INDICATOR' UPON WRKSTN.
56                 ACCEPT LIVE-INDIC FROM WRKSTN.
57                 MOVE LIVE-INDIC TO INDICATOR.
58                 DISPLAY 'INDICATOR IS SET TO ' , INDICATOR UPON WRKSTN.
59                 IF LIVE-INDIC = 3 THEN
60                   DISPLAY 'FINISHED WITH THIS #OUT STATEMENT.' UPON WRKSTN
61                   DISPLAY 'WILL PROCESS NEXT #OUT OR END.' UPON WRKSTN.
62                 DISPLAY 'EXITING OWN CODE' UPON WRKSTN.
63             OUT-EXIT.
64                 EXIT PROGRAM.
65     /*
66     // DVC RES  // LBL $Y$LOD  // LFD LOD
67     // WORK1
68     // WORK2
69     // EXEC LNKEDT
70     // PARAM OUT=LOD
71     /$
72               LOADM  ZOWNCD
73               INCLUDE MYOWN
74     /*
75     /&
```

Figure 3–6. ZOWNCD Own-Code Routine Using Spool File Input and MAPLOD Indicators (Part 2 of 2)

The linkage section in line 30 defines data transferred to MAPLOD. The items of data transferred to MAPLOD are defined in lines 31–38. The item, PRINT-IMAGE, on line 38 allows you to adjust the line length to fit the record size of the record sent to MAPLOD.

In line 42, the procedure division uses the linkage section group item, OWN-DATA. Lines 49–51 move mode number, type, and line length to the linkage section. Line 57 moves the indicator setting received from the workstation to the linkage section. When print image line processing is complete, the EXIT PROGRAM statement exits from the own-code routine.

Lines 66–74 show the link edit step to create the own-code load module that must be placed in the same load library where MAPLOD resides. In this example, $Y$LOD is the load library.

The TESTLOD4 job uses the same spool file as the previous example. In addition, it uses a MIRAM file as input to MAPLOD (lines 3 and 4) and, in line 8, it uses a different own-code routine (YOWNCD) (Figure 3–8).

```
 1     // JOB TESTLOD4,,15000
 2     // DVC 20   // LFD PRNTR
 3     // DVC 50  // VOL MAPPER
 4     // LBL MPLDTAIN  // LFD MPLIN
 5     // EXEC MAPLOD
 6     /$
 7     #INP FORM=FORMA,JOBN=PRTOUT
 8     #OWN YOWNCD
 9     #FLD 1,2,4,1;4,6,1,2;11,2,4,3
10     #FLD 14,4,4,8;19,6,1,5;26,5,1,7
11     #FLD 32,9,4,4;42,6,1,9;49,6,1,10
12     #FLD 56,6,1,11;63,5,4,12
13     #OUT JDOE,7,102,,B,120
14     /*
15     /&
```

Figure 3–7 shows an editor display of the MIRAM input file used by TESTLOD4 and the YOWNCD own code (Figure 3–8).

```
 1      ST820602JHSUNGLASS173273201121BEST830602830929830930J2061
 2      AL811002SGSUNGLASS173273301122BACH821222830121830122J2062
 3      ST820704KCSUNGLASS173273401123BACH811118112118112 12J2083
 4      EZ820801RRSUNGLASS473875602212HARK831214831215831217J2084
 5      AL820404RXSUNGLASS974231202225BEST820406820410820430J2005
 6      ST820602JHSUNGLASS173273201121BEST830602830929830930J2066
 7      AL811002SGEYEGLASS173273301122BACH821222830121830122J2067
 8      ST820704KCEYEGLASS173273401123BACH811118112118112 12T2088
 9      EZ820801RREYEGLASS473875602212HARK831214831215831217T2089
10      ST820602JHEYEGLASS173273201121BEST830602830929830930T2060
```

Figure 3–7. Editor Display of MIRAM Input File Used by TESTLOD4

The own-code routine (Figure 3–8) obtains records from both the spool file and MIRAM data file.

```
// JOB MPLDOWN2
// DVC 20  // LFD PRNTR
// WORK1
// WORK2
// WORK3
// EXEC COBL74
/$
           IDENTIFICATION DIVISION.
           PROGRAM-ID.  ANOWN.
          *REMARKS.     THIS PROGRAM ACTS AS A SUBROUTINE TO
          *             A MAPLOD JOB USED TO CREATE A MAPPER
          *             REPORT FROM A SPOOLFILE. DATA FROM A
          *             USER MIRAM FILE IS ALSO INPUT TO THE
          *             MAPPER REPORT BEING CREATED.
          * * * * * * * * * * * * *  * * * * * * * * * * * *
          *
           ENVIRONMENT DIVISION.
           CONFIGURATION SECTION.
           SOURCE-COMPUTER. UNIVAC-OS3.
           OBJECT-COMPUTER. UNIVAC-OS3.

           INPUT-OUTPUT SECTION.
           FILE-CONTROL.
               SELECT MPLDTAIN
               ASSIGN TO DISK-MPLIN-F.
          *
          * * * * * * * * * * * * *  * * * * * * * * * * * *
          *
           DATA DIVISION.
           FILE SECTION.
           FD  MPLDTAIN
               LABEL RECORD IS STANDARD.
```

Figure 3–8. YOWNCD Own-Code Routine Using Spool and MIRAM Input Files (Part 1 of 3)

```
       01  INPUT-RECORD.
           05  STATUS-CODE-IN          PIC X(2).
           05  STATUS-DATE-IN          PIC 9(6).
           05  BY-IN-IN                PIC X(2).
           05  PRODUCT-TYPE-IN         PIC X(9).
           05  SERIAL-NUMBER-IN        PIC 9(6).
           05  ORDER-NUMBER-IN         PIC 9(5).
           05  CUSTOMER-CODE-IN        PIC X(4).
           05  PRODUCTION-PLAN-IN      PIC 9(6).
           05  PRODUCTION-ACTUAL-IN    PIC 9(6).
           05  SHIPPING-DATE-IN        PIC 9(6).
           05  SHIPPING-ORDER-IN       PIC X(5).
           05  FILLER                  PIC X(199).
       WORKING-STORAGE SECTION.
       01  PROGRAM-INDICATOR.
           05  REC-CNT                 PIC 9(3) VALUE ZERO.
           05  OPN-FIL                 PIC XXX VALUE 'YES'.
           05  CLS-FIL                 PIC XXX VALUE ' NO'.
       01  WORK-RECORD.
           05  STATUS-CODE             PIC X(2) VALUE SPACES.
           05  FILLER                  PIC X VALUE SPACE.
           05  STATUS-DATE             PIC 9(6) VALUE ZEROS.
           05  FILLER                  PIC X VALUE SPACE.
           05  BY-IN                   PIC X(2) VALUE SPACES.
           05  FILLER                  PIC X VALUE SPACE.
           05  CUSTOMER-CODE           PIC X(4) VALUE SPACES.
           05  FILLER                  PIC X VALUE SPACE.
           05  SERIAL-NUMBER           PIC 9(6) VALUE ZEROS.
           05  FILLER                  PIC X VALUE SPACE.
           05  ORDER-NUMBER            PIC 9(5) VALUE ZEROS.
           05  FILLER                  PIC X VALUE SPACE.
           05  PRODUCT-TYPE            PIC X(9) VALUE SPACES.
           05  FILLER                  PIC X VALUE SPACE.
           05  PRODUCTION-PLAN         PIC 9(6) VALUE ZEROS.
           05  FILLER                  PIC X VALUE SPACE.
           05  PRODUCTION-ACTUAL       PIC 9(6) VALUE ZEROS.
           05  FILLER                  PIC X VALUE SPACE.
           05  SHIPPING-DATE           PIC 9(6) VALUE ZEROS.
           05  FILLER                  PIC X VALUE SPACE.
           05  SHIPPING-ORDER          PIC X(5) VALUE SPACES.
       LINKAGE SECTION.
       01  OWN-DATA.
           02  MODE-NO                 PIC 9(3).
           02  TYPEX                   PIC X.
           02  RID-NO                  PIC 9(3).
           02  INDICATOR               PIC 9.
           02  IMAGE-LENGTH            PIC 9(4)  COMP.
           02  CONTROL-CODE            PIC X.
           02  PRINT-IMAGE             PIC X(90).
      *
      * * * * * * * * * * * *  * * * * * * * * * * * *
      *
       PROCEDURE DIVISION       USING OWN-DATA.
      *
      * * * * * * * * * * * *  * * * * * * * * * * * *
      *
```

Figure 3–8. YOWNCD Own-Code Routine Using Spool and MIRAM Input Files (Part 2 of 3)

```
           MAIN-LINE.
               IF OPN-FIL = 'YES' THEN PERFORM OPEN-FILE-RTN.
               COMPUTE REC-CNT = REC-CNT + 1.
               IF REC-CNT < 11 THEN PERFORM PROCESSING-LOOP.
               IF CLS-FIL = 'YES' THEN PERFORM CLOSE-FILE-RTN.
               MOVE 102 TO MODE-NO.
               MOVE 'B' TO TYPEX.
               MOVE 90 TO IMAGE-LENGTH.
               IF REC-CNT < 10 THEN MOVE 1 TO INDICATOR.
               IF REC-CNT > 9 THEN MOVE 0 TO INDICATOR.
           OUT-EXIT.
               EXIT PROGRAM.
          *
          * * * * * * * * * * * *  * * * * * * * * * * * *
          *
           OPEN-FILE-RTN.
               OPEN INPUT MPLDTAIN.
               MOVE ' NO' TO OPN-FIL.
          *
          * * * * * * * * * * * *  * * * * * * * * * * * *
          *
           PROCESSING-LOOP.
               READ MPLDTAIN.
               IF REC-CNT = 10 THEN MOVE 'YES' TO CLS-FIL.
               MOVE STATUS-CODE-IN TO STATUS-CODE.
               MOVE STATUS-DATE-IN TO STATUS-DATE.
               MOVE BY-IN-IN TO BY-IN.
               MOVE PRODUCT-TYPE-IN TO PRODUCT-TYPE.
               MOVE SERIAL-NUMBER-IN TO SERIAL-NUMBER.
               MOVE ORDER-NUMBER-IN TO ORDER-NUMBER.
               MOVE CUSTOMER-CODE-IN TO CUSTOMER-CODE.
               MOVE PRODUCTION-PLAN-IN TO PRODUCTION-PLAN.
               MOVE PRODUCTION-ACTUAL-IN TO PRODUCTION-ACTUAL.
               MOVE SHIPPING-DATE-IN TO SHIPPING-DATE.
               MOVE SHIPPING-ORDER-IN TO SHIPPING-ORDER.
               MOVE WORK-RECORD TO PRINT-IMAGE.
          *
          * * * * * * * * * * * *  * * * * * * * * * * * *
          *
           CLOSE-FILE-RTN.
               CLOSE MPLDTAIN.
               MOVE ' NO' TO CLS-FIL.
/*
// DVC RES  // LBL $Y$LOD  // LFD LOD
// WORK1
// WORK2
// EXEC LNKEDT
// PARAM OUT=LOD
/$
        LOADM  YOWNCD
        INCLUDE ANOWN
/*
/&
```

Figure 3–8. YOWNCD Own-Code Routine Using Spool and MIRAM Input Files (Part 3 of 3)

INPUT-RECORD defined on lines 33–45 describes the record layout of the MIRAM file. YOWNCD moves data read from the MIRAM file to the WORK-RECORD area in working-storage (lines 51–72). Before moving this data to the WORK-RECORD (lines 113–123), your own code could also perform internal calculations and send the results to WORK-RECORD in working-storage.

The WORK-RECORD layout matches the #FLD statements in the MAPLOD execution stream (TESTLOD4) because own code moves the WORK-RECORD to the linkage section (line 124). This record replaces the PRINT-IMAGE line in the print buffer, and MAPLOD verifies it as a print image line.

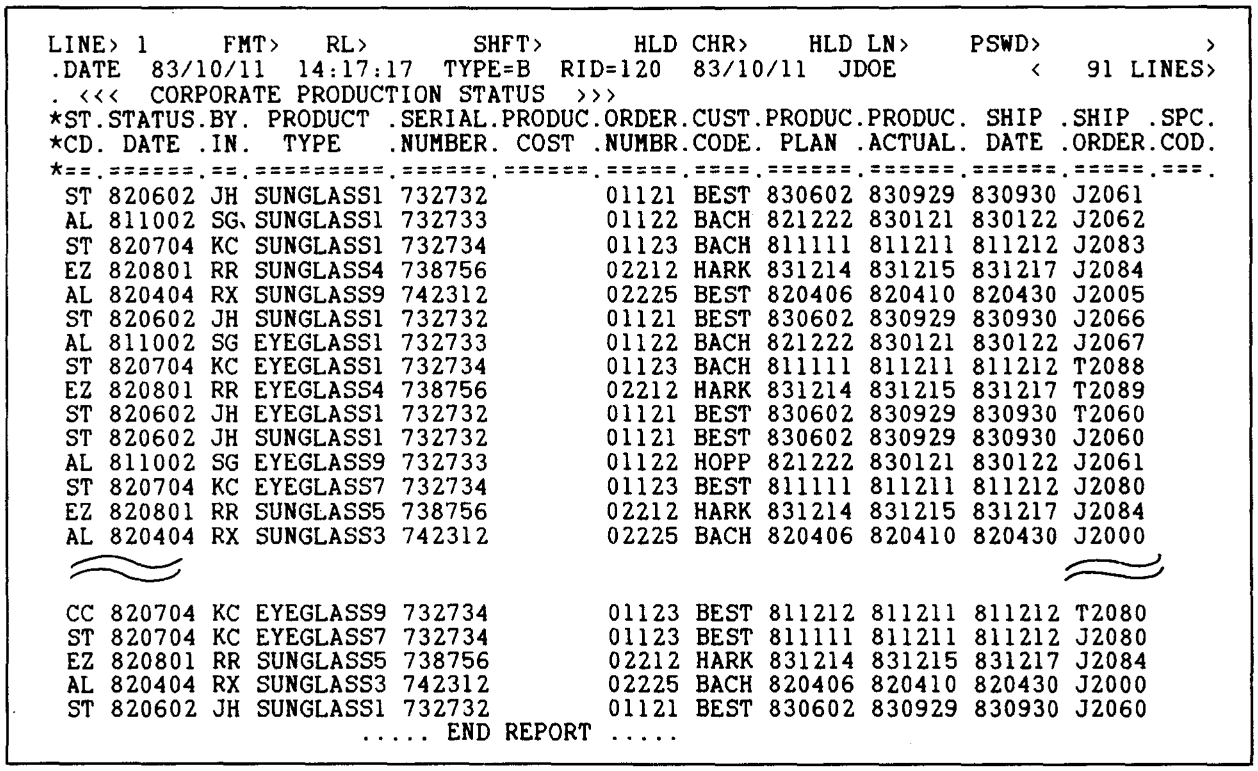
Own-code logic sets the indicators (lines 97 and 98). Ten records are in the input MIRAM file, and the own code uses a counter to determine the indicator settings (lines 92, 97, and 98).

The own code first sends the 10 MIRAM records followed by the 75 PRINT-IMAGE records (lines 92 and 110–124) to the MAPPER 80 report. After own code reads the 10 records from the MIRAM file, it closes the file (lines 93 and 112). Then, own code sets the indicator to 0 and directs the print image records to the MAPPER 80 report (line 124).

The linkage section defines, in a single group item (lines 74–81), the data transferred to MAPLOD. In line 85, the procedure division uses the linkage section group item, OWN-DATA. When processing is complete, the EXIT PROGRAM statement (lines 99–100) exits from the own-code routine.

Lines 132–140 show the link edit step to create the own-code load module that must be placed in the same load library where MAPLOD resides. In this example, $Y$LOD is the load library.

Figure 3–9 is a MAPPER 80 display of the resulting RID (120B) created by TESTLOD4.



```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>     PSWD>           >
.DATE  83/10/11  14:17:17  TYPE=B  RID=120  83/10/11  JDOE          <    91 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 ST 820602 JH SUNGLASS1 732732        01121 BEST 830602 830929 830930 J2061
 AL 811002 SG SUNGLASS1 732733        01122 BACH 821222 830121 830122 J2062
 ST 820704 KC SUNGLASS1 732734        01123 BACH 811111 811211 811212 J2083
 EZ 820801 RR SUNGLASS4 738756        02212 HARK 831214 831215 831217 J2084
 AL 820404 RX SUNGLASS9 742312        02225 BEST 820406 820410 820430 J2005
 ST 820602 JH SUNGLASS1 732732        01121 BEST 830602 830929 830930 J2066
 AL 811002 SG EYEGLASS1 732733        01122 BACH 821222 830121 830122 J2067
 ST 820704 KC EYEGLASS1 732734        01123 BACH 811111 811211 811212 T2088
 EZ 820801 RR EYEGLASS4 738756        02212 HARK 831214 831215 831217 T2089
 ST 820602 JH EYEGLASS1 732732        01121 BEST 830602 830929 830930 T2060
 ST 820602 JH SUNGLASS1 732732        01121 BEST 830602 830929 830930 J2060
 AL 811002 SG EYEGLASS9 732733        01122 HOPP 821222 830121 830122 J2061
 ST 820704 KC EYEGLASS7 732734        01123 BEST 811111 811211 811212 J2080
 EZ 820801 RR SUNGLASS5 738756        02212 HARK 831214 831215 831217 J2084
 AL 820404 RX SUNGLASS3 742312        02225 BACH 820406 820410 820430 J2000

 CC 820704 KC EYEGLASS9 732734        01123 BEST 811212 811211 811212 T2080
 ST 820704 KC EYEGLASS7 732734        01123 BEST 811111 811211 811212 J2080
 EZ 820801 RR SUNGLASS5 738756        02212 HARK 831214 831215 831217 J2084
 AL 820404 RX SUNGLASS3 742312        02225 BACH 820406 820410 820430 J2000
 ST 820602 JH SUNGLASS1 732732        01121 BEST 830602 830929 830930 J2060
                         ..... END REPORT .....
```

Figure 3–9. Display of RID 120B Created by TESTLOD4 MAPLOD Execution with YOWNCD

3.5. MAPLOD ERROR MESSAGES

Table 3–4 lists MAPLOD error messages.

Table 3–4. MAPLOD Error Messages

Error Message Number	Error Message Text
MPL11	OPEN ERROR (SPOOL IMAGE FILE)
MPL12	OPEN ERROR (MAPPER DATA BASE)
MPL21	MISSING INPUT PARAMETER
MPL22	OWN CODE LOAD ERROR
MPL23	NO #INP CARD
MPL24	NO FORM= PARAMETER IN #INP CARD
MPL25	INVALID #INP CARD
MPL31	PARAMETER IS NOT #FLD CARD
MPL32	ILLEGAL LOCATION IN #FLD
MPL33	ILLEGAL LENGTH IN #FLD
MPL34	ILLEGAL FORMAT IN #FLD
MPL35	ILLEGAL ITEM-NUMBER IN #FLD
MPL41	PARAMETER IS NOT #OUT CARD
MPL42	ILLEGAL USER-ID IN #OUT
MPL43	ILLEGAL DEPARTMENT-NUMBER IN #OUT
MPL44	ILLEGAL MODE-NUMBER IN #OUT
MPL45	ILLEGAL PASSWORD IN #OUT
MPL46	ILLEGAL TYPE-NUMBER IN #OUT
MPL47	ILLEGAL REPORT-ID IN #OUT
MPL48	ILLEGAL START PAGE# IN #OUT
MPL49	ILLEGAL END PAGE# IN #OUT
MPL50	ILLEGAL EJECT-CODE IN #OUT
MPL51	REQUESTED USER-ID DOES NOT EXIST
MPL52	REQUESTED MODE DOES NOT EXIST
MPL53	INVALID MODE PASSWORD
MPL54	RID-Ø DOES NOT EXIST
MPL55	RID-Ø IS INCORRECT
MPL56	ITEM-NUMBER(nn) DOES NOT EXIST
MPL57	FIELD-FORMAT IS UNMATCHED (nn)
MPL58	FREE RID DOES NOT EXIST IN THE TYPE
MPL59	REQUESTED RID DOES NOT EXIST
MPL71	LINE# OF OUTPUT REPORT IS OVER THE LIMIT
MPL72	ILLEGAL DATA RETURNED FROM USER OWN CODE
MPL73	READ ERROR (SPOOL IMAGE FILE)
MPL74	SPOOL IMAGE RECORD IS INCORRECT
MPL81	ACCESS OF DMS HAS DEAD-LOCK, RETRY
MPL82	LACK OF DMS SPACE, EXPAND THE SPACE
MPL85	MAPLOD HAS FAILED
MPL86	MPDP HAS FAILED
MPL90	PROGRAM CHECK OCCURRED IN OWN CODE
MPL96	LAST ROLL BACK HAS FAILED
MPL97	ABNORMAL COMPLETION
MPL98	REPORT COMPLETE (nnnn LINES)
MPL99	NORMAL COMPLETION

System error codes that may occur during MAPLOD execution:

998 – Program check occurred in user own code.
999 – DMS error other than deadlock or no space occurred in MAPLOD.

4. MAPLST Utility

4.1. WHAT THE MAPLST UTILITY DOES

MAPLST is a batch utility program that prints MAPPER 80 reports. It is similar to the PRINT (PR) manual function used in the MAPPER 80 interactive environment. Figure 4–1 shows a control flow chart of the MAPLST utility.

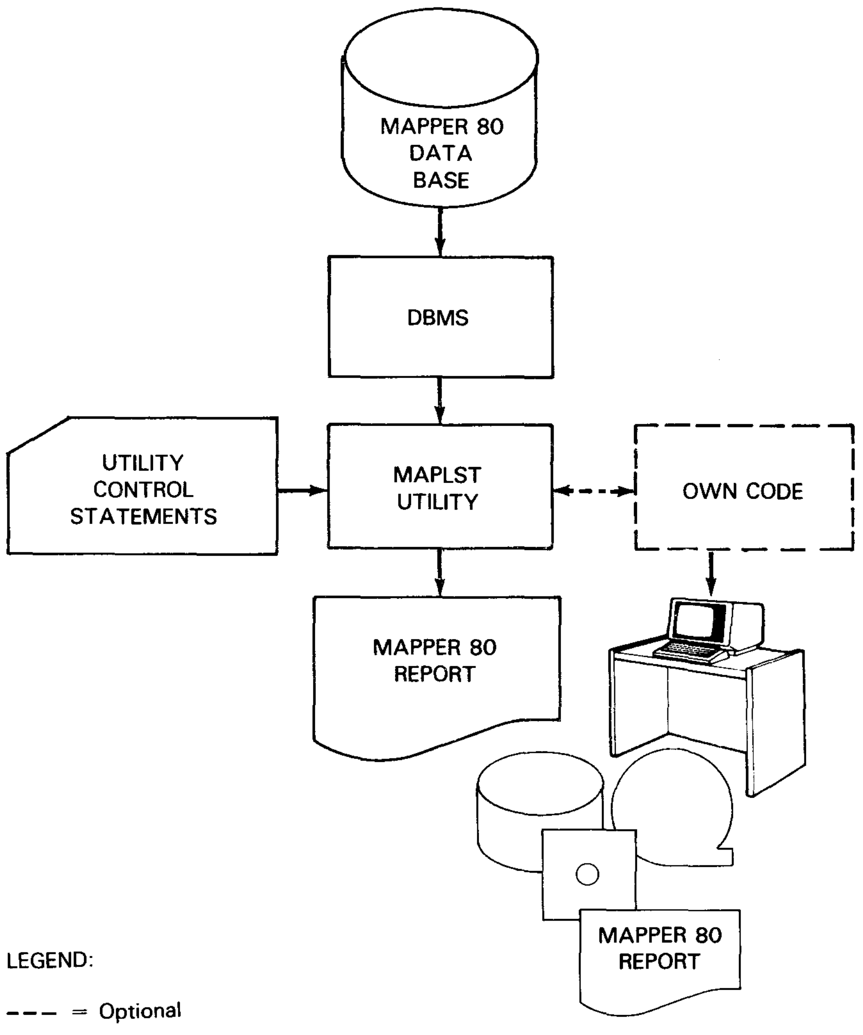


Figure 4–1. Control Flow for MAPLST Utility Execution

The MAPLST utility performs the following:

- Sends MAPPER 80 reports from the data base to the printer
- Prints multiple reports in one execution
- Specifies the first column of an output line. This is similar to the interactive modified format for the PR manual function.
- Places MAPPER 80 report header lines at the beginning of each page of a report
- Employs your own code to process MAPPER 80 reports; however, your own code cannot modify the print content or update the input report.

The MAPLST utility assumes the page size to be 66 lines per page. Fifty of the 66 lines are printed lines with a margin at the top and bottom. The MAPLST utility starts printing a new page under two conditions:

- When there are more than 50 lines in the report to be printed, MAPLST executes a top of form because it only prints 50 lines per page.
- When MAPLST encounters an .EJECT in the job control stream

4.2. MAPLST CONTROL STATEMENTS

The MAPLST utility control statements designate attributes of the report printed by MAPLST, your own code, printing of report header lines, line spacing, etc, using the following five control statements:

#OWN	Indicates the linkage and release of your own code
#HDL	Indicates the number of header lines
#SPC	Indicates the line spacing on the output report
#LST	Specifies which report is to be printed
#END	Specifies the termination of the MAPLST utility

4.2.1. Including Your Own Code in MAPLST Utility (#OWN)

The #OWN control statement indicates the linkage and release of your own code and MAPLST utility. Use this statement only when employing your own code. Use multiple #OWN statements to change to another own-code routine during execution. Use the statement anywhere in the control stream as long as it precedes the #END statement. The new own code takes effect on the next appearance of a #LST statement.

Format:

```
    #OWN phasename
```

where:

`phasename`
: Identifies the phasename provided at link time for your own code, a loadable program that was compiled and linked in a separate job.

  If you omit phasename and your own code is released in the MAPLST execution, your own code used to that point is ignored.

  If your own code is changed or released, the own code used to that point cannot be released from main storage.

  NOTE:

  When you specify the amount of main storage allocation to the job on the JOB job control statement, include the main storage needed for your own code.

4.2.2. Specifying Continuous Headings on MAPPER 80 Reports (#HDL)

The #HDL control statement specifies continued printing of MAPPER 80 report header lines on and after the second page of the report being printed.

You can use multiple statements anywhere before the #END control statement. Header lines are printed when the next #LST statement appears.

Format:

```
    #HDL n
```

where:

n

Specifies 0 to 5 header lines. Header lines following line 1 of the report appear at the top of each page. You specify how many of these lines are to be printed (0–5).

Specifying 0 causes the previous number of lines specification to be ignored, and headers occur only on the first page.

When MAPLST executes, the number of lines is set to 0. If you omit the #HDL control statement, headers only appear on the first page. Lines designated by the #HDL statement are single-spaced when printed. Note that #SPC control statement specifications (4.2.3) have no effect on these header lines.

4.2.3. Controlling Line Spacing on MAPPER 80 Printed Reports (#SPC)

The #SPC control statement determines the line spacing for the printed report. The default is single-spacing.

You can use multiple #SPC statements anywhere before the #END control statement. The desired data is printed when the first #LST control statement appears.

Format:

```
#SPC n
```

where:

n

Specifies the space between lines is from 1 to 9 (i.e., double-spacing is 2 and single-spacing is 1). Lines designated by the #HDL statement are not affected by this control statement.

4.2.4. Specifying the Report for Printout (#LST)

The #LST control statement specifies the report to be printed and the print position (column number) in the line image of the report.

The #LST statement may appear more than once in the job control stream as long as it appears before the #END statement.

Execution of the #OWN, #HDL, and #SPC statements preceding the #LST statement begins when the #LST statement is encountered.

Format:

```
#LST mmmtrrr[sss][eee]
```

where:

`mmm`
: Specifies the mode number of the report to be printed; both even and odd modes are available. This value must be 3 digits; leading zeros are required, e.g., 4 is 004.

`t`
: Indicates the type of the report to be printed (A–I).

`rrr`
: Is the report or RID number. This value must be 3 digits, and leading zeros are required.

`sss`
: Indicates the first column number in the data line of the printed report. Specify this parameter using three digits with leading zeros. The default value is 1.

`eee`
: Is the last column in the data line of the report to be printed. Three digits must be specified, and leading zeros are required.

4.2.5. Terminating MAPLST Utility Processing (#END)

The #END control statement terminates the MAPLST utility processing. The #END statement must appear at the end of the job control stream.

4.3. MAPLST EXECUTION

When you are ready to print MAPPER 80 reports, you use the following job control stream to execute the MAPLST utility:

```
// JOB jobname
// DVC 2Ø   // LFD PRNTR
     (include other device assignment sets here)
// EXEC MAPLST
/$
     (utility control statements)
/*
/&
// FIN
```

Remember, the DBMS job must be started before the MAPLST utility executes.

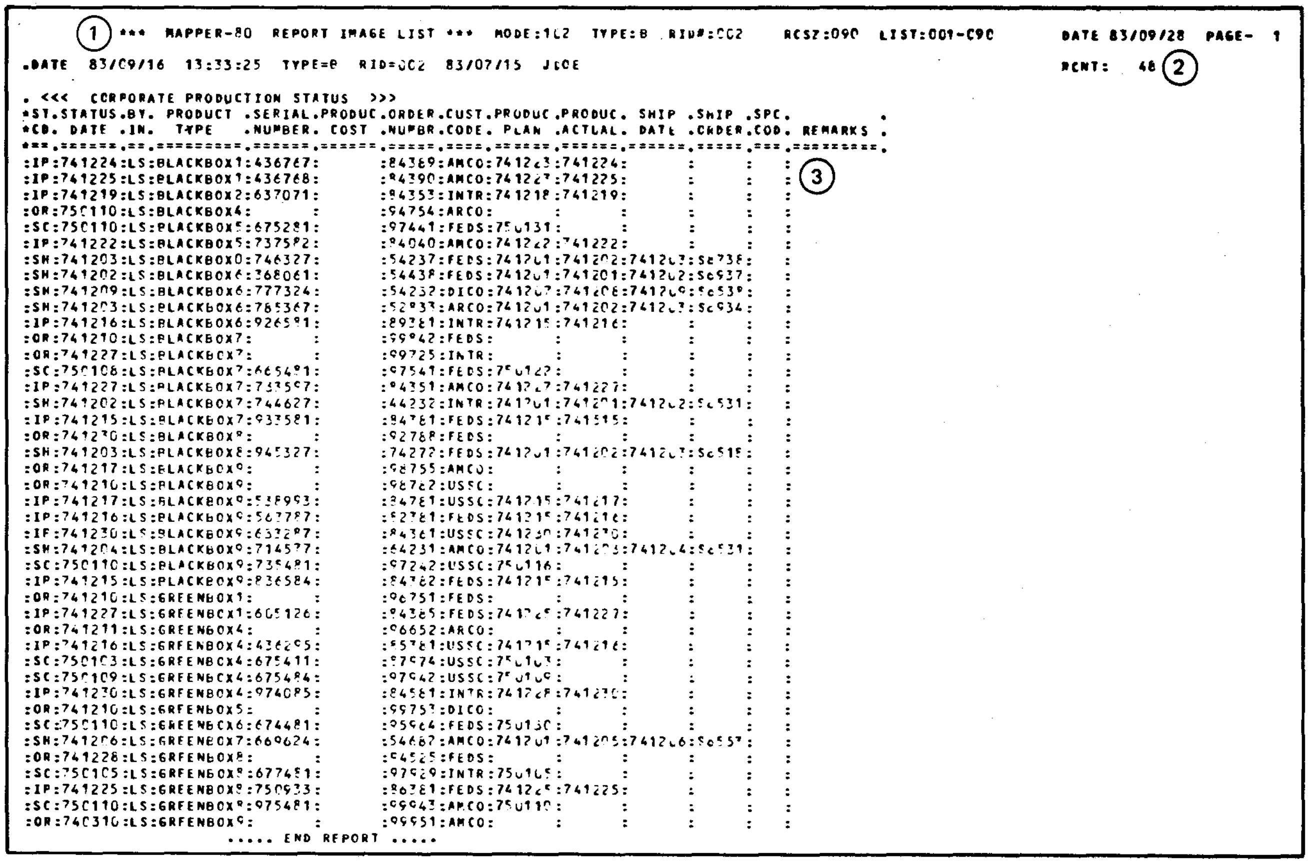
The following sample job control stream shows the statements needed for a simple execution of the MAPLST utility that produces a listing of a MAPPER 80 report (RID 2B in mode 102).

NOTE:

MAPPER 80 reports being processed in the following MAPLST sample job streams are contained in the JDOE training data base delivered with the MAPPER 80 system.

```
// JOB TESTLST1
// DVC 20  // LFD PRNTR
// EXEC MAPLST
/$
#HDL 4
#LST 102B002
#END
/*
/&
```

Figure 4-2 shows the listing that results from the previous sample execution of MAPLST.



```
  ① *** MAPPER-80  REPORT IMAGE LIST ***  MODE:102  TYPE:B  RID#:002   RCSZ:090  LIST:001-090        DATE 83/09/28  PAGE-  1
.DATE  83/09/16  13:33:25  TYPE=B  RID=002  83/07/15  JDOE                                            RCNT:   48 ②

. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.          .
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD. REMARKS .
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.=========.
:IP:741224:LS:BLACKBOX1:436767:      :84369:AMCO:741223:741224:      :     :   : ③
:IP:741225:LS:BLACKBOX1:436768:      :84390:AMCO:741223:741225:      :     :   :
:IP:741219:LS:BLACKBOX2:637071:      :84353:INTR:741218:741219:      :     :   :
:OR:750110:LS:BLACKBOX4:      :      :94754:ARCO:      :      :      :     :   :
:SC:750110:LS:BLACKBOX5:675281:      :97441:FEDS:750131:      :      :     :   :
:IP:741222:LS:BLACKBOX5:737582:      :84040:AMCO:741222:741222:      :     :   :
:SH:741203:LS:BLACKBOX0:746327:      :54237:FEDS:741201:741202:741203:S0738:   :
:SH:741202:LS:BLACKBOX6:368061:      :54438:FEDS:741201:741201:741202:S0937:   :
:SH:741209:LS:BLACKBOX6:777324:      :54232:DICO:741207:741208:741209:S0538:   :
:SH:741203:LS:BLACKBOX6:785367:      :52833:ARCO:741201:741202:741203:S0934:   :
:IP:741216:LS:BLACKBOX6:926581:      :89381:INTR:741215:741216:      :     :   :
:OR:741210:LS:BLACKBOX7:      :      :99842:FEDS:      :      :      :     :   :
:OR:741227:LS:BLACKBOX7:      :      :99725:INTR:      :      :      :     :   :
:SC:750108:LS:BLACKBOX7:665481:      :97541:FEDS:750122:      :      :     :   :
:IP:741227:LS:BLACKBOX7:735597:      :84351:AMCO:741227:741227:      :     :   :
:SH:741202:LS:BLACKBOX7:744627:      :44232:INTR:741201:741201:741202:S0531:   :
:IP:741215:LS:BLACKBOX7:937581:      :84781:FEDS:741215:741515:      :     :   :
:OR:741230:LS:BLACKBOX8:      :      :92788:FEDS:      :      :      :     :   :
:SH:741203:LS:BLACKBOX8:945327:      :74272:FEDS:741201:741202:741203:S0515:   :
:OR:741217:LS:BLACKBOX9:      :      :98755:AMCO:      :      :      :     :   :
:OR:741210:LS:BLACKBOX9:      :      :98782:USSC:      :      :      :     :   :
:IP:741217:LS:BLACKBOX9:538993:      :84781:USSC:741215:741217:      :     :   :
:IP:741216:LS:BLACKBOX9:567787:      :82781:FEDS:741215:741216:      :     :   :
:IP:741230:LS:BLACKBOX9:633287:      :84361:USSC:741230:741230:      :     :   :
:SH:741204:LS:BLACKBOX9:714577:      :64231:AMCO:741201:741203:741204:S0531:   :
:SC:750110:LS:BLACKBOX9:735481:      :97242:USSC:750116:      :      :     :   :
:IP:741215:LS:BLACKBOX9:836584:      :84782:FEDS:741215:741215:      :     :   :
:OR:741210:LS:GREENBOX1:      :      :96751:FEDS:      :      :      :     :   :
:IP:741227:LS:GREENBOX1:605126:      :84385:FEDS:741225:741227:      :     :   :
:OR:741211:LS:GREENBOX4:      :      :96652:ARCO:      :      :      :     :   :
:IP:741216:LS:GREENBOX4:436295:      :85781:USSC:741215:741216:      :     :   :
:SC:750103:LS:GREENBOX4:675411:      :97974:USSC:750103:      :      :     :   :
:SC:750109:LS:GREENBOX4:675484:      :97942:USSC:750109:      :      :     :   :
:IP:741230:LS:GREENBOX4:974085:      :84581:INTR:741228:741230:      :     :   :
:OR:741210:LS:GREENBOX5:      :      :99753:DICO:      :      :      :     :   :
:SC:750110:LS:GREENBOX6:674481:      :95964:FEDS:750130:      :      :     :   :
:SH:741206:LS:GREENBOX7:669624:      :54687:AMCO:741201:741205:741206:S0557:   :
:OR:741228:LS:GREENBOX8:      :      :54525:FEDS:      :      :      :     :   :
:SC:750105:LS:GREENBOX8:677451:      :97929:INTR:750105:      :      :     :   :
:IP:741225:LS:GREENBOX8:750933:      :86381:FEDS:741225:741225:      :     :   :
:SC:750110:LS:GREENBOX8:975481:      :99943:AMCO:750110:      :      :     :   :
:OR:740310:LS:GREENBOX9:      :      :99951:AMCO:      :      :      :     :   :
                    ..... END REPORT .....
```

NOTES:

① MAPLST headers are printed on each page.

② Line 1 and the report password and the number of lines appear at the top of each page.

③ Colons (:) are substituted for the tabs between the data columns, and are printed.

Figure 4-2. Sample Report Listing (RID 2B, Mode 102) after MAPLST Execution

The following sample job control stream contains all of the previously discussed MAPLST utility control statements:

```
 1       // JOB TESTLST2,,28000
 2       // DVC 20  // LFD PRNTR
 3       // DVC 50  // VOL MAPPER
 4       // LBL MPLSTOUT  // LFD MPLSOT
 5       // SFT DLOAD=(1,4000)
 6       // EXEC MAPLST
 7       /$
 8       #OWN UOWNCD00
 9       #HDL 4
10       #LST 102B002
11       #OWN
12       #SPC 2
13       #HDL 0
14       #LST 102D001011048
15       #END
16       /*
17       /&
```

NOTE:

This MAPLST job control stream may be used with the own-code sample shown in Figure 4–5.

Line 1 designates the size of main storage including your own code. Next, the device assignment set (line 2) assigns the printer. PRNTR must be used as the LFD name. Lines 3 and 4 assign the output MIRAM file used in this job stream and by the sample own code (Figure 4–5). This MIRAM file must be previously allocated. Line 5 assigns the main storage area for dynamic loading of your own code. The number of calls is one and the // SFT statements request expansion for X'4000' bytes.

The first MAPLST control statement (line 8) states that your own code with the phase name UOWNCD is prepared. When the #LST control statement executes in line 10, control transfers to your own code.

The #HDL control statement (line 9) indicates that four lines, beginning with the second line of the report, are printed on each page. These lines are usually for headers in a MAPPER 80 report. When the #LST control statement executes (line 10), this header control takes effect.

Line 10 states that a mode 102, type B, RID 2 report will be printed. Line spacing is 1, and 4 lines of headers print on each page.

The #SPC control statement (line 12) indicates that the line spacing changes to double-spacing when the next report begins printing.

The second #HDL statement (line 13) indicates header line print termination. This is effective when the #LST control statement in line 14 executes.

Line 11 signals the termination of transfer of control to your own code. This termination is effective when the #LST control statement (line 14) executes. Until then, control is transferred to your own code when each single line of the report is read in.

The #LST statement (line 14) signifies that a mode 102, type D, RID 1 report is printed. Starting with column 11 of the report, 138 columns are printed. Line spacing is double-space; after the second page of the report, no headers are printed.

Executing MAPLST for this TESTLST2 job produces a:

- Listing of RID 2B from mode 102 (Figure 4–2)
- Listing of RID 1D from mode 102 (Figure 4–3)
- MIRAM file (Figure 4–4) created by your own code (Figure 4–5)

```
         *** MAPPER-80  REPORT IMAGE LIST ***  MODE:1C2  TYPE:D  RID#:001    RCSZ:128  LIST:011-048       DATE 83/09/28  PAGE-  1

.DATE  83/C7/15  11:04:53  TYPE=D  RID=001  83/07/15  JLOE                                              RCNT:   20

PORATE ORDER STATUS  >>>
. PRODUCT .ODR.CUST.  UNIT  .EXTENDED.
.  TYPE   .QTY.CODE. RETAIL . RETAIL .
.=========.===.====.========.========.

:GREENBOX9:  2:AMCO:        :        :

:GREENBOX7:  1:AMCO:        :        :

:BLACKBOX9:  1:AMCO:        :        :

:GREENBOX4:  2:ARCO:        :        :

:BLACKBOX5:  1:ARCO:        :        :

:BLACKBOX4:  1:ARCO:        :        :

:GREFNBOX5:  1:DICO:        :        :

:GREENBOX8:  1:FFDS:        :        :

:GREENBOX1:  1:FEDS:        :        :

:BLACKBOX8:  1:FEDS:        :        :

:BLACKBOX0:  1:FEDS:        :        :

:BLACKBOX4:  1:INTR:        :        :

:BLACKBOX9:  1:USSC:        :        :

:GREENBOX9:  1:USSC:        :        :

        ..... END REPORT .....
```

Figure 4–3. Listing of RID 1D Resulting from MAPLST Execution (TESTLST2)

Figure 4–4 shows an OS/3 editor display of the MIRAM file resulting from the sample MAPLST job control stream (TESTLST2) using the sample own code in Figure 4–5.

```
 1.0000 ZH 741224 BLACKBOX1 741223
 2.0000 IP 741225 BLACKBOX1 741223
 3.0000 IP 741219 BLACKBOX2 741218
 4.0000 OR 750110 BLACKBOX4
 5.0000 SC 750110 BLACKBOX5 750131
 6.0000 IP 741222 BLACKBOX5 741222
 7.0000 SH 741203 BLACKBOX0 741201 741203 S8738
 8.0000 SH 741202 BLACKBOX6 741201 741202 S6937
 9.0000 SH 741209 BLACKBOX6 741207 741209 S8538
10.0000 SH 741203 BLACKBOX6 741201 741203 S8934
11.0000 IP 741216 BLACKBOX6 741215
12.0000 OR 741210 BLACKBOX7
13.0000 OR 741227 BLACKBOX7
14.0000 SC 750108 BLACKBOX7 750122
15.0000 IP 741227 BLACKBOX7 741227
16.0000 SH 741202 BLACKBOX7 741201 741202 S8531
17.0000 IP 741215 BLACKBOX7 741215
18.0000 OR 741230 BLACKBOX8
19.0000 SH 741203 BLACKBOX8 741201 741203 S8518
20.0000 OR 741217 BLACKBOX9
```

Figure 4–4. Contents of MIRAM File Created by Own Code during MAPLST Execution (TESTLST2)

4.4. YOUR OWN CODE

4.4.1. Applying Your Own Code

Several ways to apply your own code when using the MAPLST utility include:

- Divide one line in MAPPER 80 report and print it as two lines (2-stage handling).
- Copy a MAPPER 80 report to a file with a different format, e.g., output to magnetic tape or MIRAM file.

NOTE:

Your own code cannot use a DMS data base. To direct data to another DMS data base, write your own code to store data in an intermediate data file. Then, transfer the data to a DMS data base by using a separate DML application program.

4.4.2. Transferring Control to Your Own Code

Control transfers to your own code each time MAPLST obtains the MAPPER 80 report line record. MAPLST transfers one line at a time from the first line in the report to the last line.

The transferred data can be a maximum of 178 bytes. Table 4–1 describes the contents of these bytes by their byte position.

Table 4–1. MAPLST Data Transfer Description

Byte Position	Number of Bytes	Contents
1 to 3	3	Mode number of the report
4	1	Type of report
5 to 7	3	RID number of report
8	1	Not used
9 to 10	2	Line image length, set in binary. There are 80 to 132 bytes, but within one report the length is the same. Line 1 and lines preceded by a period are also transferred at tab line length.
11	1	Sets classification of the lines. ■ H Line 1 and header lines (4 lines) ■ L Last line (end of report line) ■ Other than H or L for all other lines.
12 to 46	35	Not used
47 to 178	Maximum 132	Sets line image. Line image has the same content as the image on the screen.

NOTE:

With line 1, the report password is also set (6 bytes) from the eighty-first byte of the line image. Also, at the start of a line and at its end, as well as between fields, tabs are set with the tab code (X'05'). MAPLST includes these tab sets in the report data line. Remember to remove them within your own code.

Two considerations must be made when exiting from your own code:

1. Don't change the report image within the transferred data.
2. Return control to the MAPLST utility through the EXIT PROGRAM instruction (in COBOL), because your own code is a called program.

4.4.3. Executing Your Own Code

Your own code for the MAPLST utility is prepared as a called program. This independent own code module is linked and the load module is placed in the same load library where MAPLST resides. Your own-code module is linked to MAPLST at execution time through the OS/3 dynamic load function.

MAPLST always produces a listing of the MAPPER 80 report, with or without your own code.

NOTE:

When separately compiling and linking the sample own code (Figure 4–5) to create a load module, use the sample MAPLST job stream TESTLST2.

4.4.4. Example of Your Own Code

The following is an example of your own-code placement in the framework of a COBOL program:

```
// JOB MPLSTOWN
// DVC 20  // LFD PRNTR
// WORK1
// WORK2
// WORK3
// EXEC COBL74
/$
        IDENTIFICATION DIVISION.
        PROGRAM-ID.  MYPROG.
       *REMARKS.     THIS PROGRAM ACTS AS A SUBROUTINE TO
       *             A MAPLST JOB USED TO OUTPUT A MAPPER
       *             REPORT FROM THE MAPPER DATA BASE.
       * * * * * * * * * * * *  * * * * * * * * * * * *
       *
        ENVIRONMENT DIVISION.
        CONFIGURATION SECTION.
        SOURCE-COMPUTER. UNIVAC-OS3.
        OBJECT-COMPUTER. UNIVAC-OS3.

        INPUT-OUTPUT SECTION.
        FILE-CONTROL.
            SELECT MPLSTOUT
            ASSIGN TO DISK-MPLSOT-F.
       *
       * * * * * * * * * * * *  * * * * * * * * * * * *
       *
        DATA DIVISION.
        FILE SECTION.
        FD  MPLSTOUT
            LABEL RECORD IS STANDARD.
        01  OUTPUT-RECORD.
            05  STATUS-CODE             PIC X(2).
            05  FILLER                  PIC X.
            05  STATUS-DATE             PIC 9(6).
            05  FILLER                  PIC X.
            05  PRODUCT-TYPE            PIC X(9).
            05  FILLER                  PIC X.
            05  PRODUCTION-PLAN         PIC 9(6).
            05  FILLER                  PIC X.
            05  SHIPPING-DATE           PIC 9(6).
            05  FILLER                  PIC X.
            05  SHIPPING-ORDER          PIC X(5).
        WORKING-STORAGE SECTION.
        01  PROGRAM-INDICATOR.
```

Figure 4–5. Sample of User Own Code (Part 1 of 3)

```
         05  OPN-FIL                  PIC XXX VALUE 'YES'.
         05  CLS-FIL                  PIC XXX VALUE ' NO'.
         05  REC-CNT                  PIC 9(3) VALUE ZEROS.
     01  WORK-RECORD.
         05  FILLER                PIC X.
         05  STATUS-CODE-WK        PIC X(2).
         05  FILLER                PIC X.
         05  STATUS-DATE-WK        PIC 9(6).
         05  FILLER                PIC X.
         05  BY-IN-WK              PIC X(2).
         05  FILLER                PIC X.
         05  PRODUCT-TYPE-WK       PIC X(9).
         05  FILLER                PIC X.
         05  SERIAL-NUMBER-WK      PIC 9(6).
         05  FILLER                PIC X.
         05  PRODUCT-COST-WK       PIC 9(6).
         05  FILLER                PIC X.
         05  ORDER-NUMBER-WK       PIC 9(5).
         05  FILLER                PIC X.
         05  CUSTOMER-CODE-WK      PIC X(4).
         05  FILLER                PIC X.
         05  PRODUCTION-PLAN-WK    PIC 9(6).
         05  FILLER                PIC X.
         05  PRODUCTION-ACTUAL-WK  PIC 9(6).
         05  FILLER                PIC X.
         05  SHIPPING-DATE-WK      PIC 9(6).
         05  FILLER                PIC X.
         05  SHIPPING-ORDER-WK     PIC X(5).
         05  FILLER                PIC X.
         05  SPECIAL-CODE-WK       PIC X(3).
         05  FILLER                PIC X.
         05  REMARKS-WK            PIC X(9).
         05  FILLER                PIC X.
     LINKAGE SECTION.
     01  OWN-DATA.
         02  MODE-NO               PIC 9(3).
         02  TYPEX                 PIC X.
         02  RID-NO                PIC 9(3).
         02  FILLER                PIC X.
         02  LINE-LENGTH           PIC 9(4)  COMP.
         02  LINE-TYPE             PIC X.
         02  FILLER                PIC X(35).
         02  LINE-IMAGE            PIC X(90).
    *
```

Figure 4–5. Sample of User Own Code (Part 2 of 3)

```
   * * * * * * * * * * * *  * * * * * * * * * * * *
   *
    PROCEDURE DIVISION       USING OWN-DATA.
   *
   * * * * * * * * * * * *  * * * * * * * * * * * *
   *
    MAIN-LINE.
        IF OPN-FIL = 'YES' THEN PERFORM OPEN-FILE-RTN.
        COMPUTE REC-CNT = REC-CNT + 1.
        IF REC-CNT > 5 AND REC-CNT < 26 THEN
            PERFORM PROCESSING-LOOP.
        IF REC-CNT = 25 THEN MOVE 'YES' TO CLS-FIL.
        IF CLS-FIL = 'YES' THEN PERFORM CLOSE-FILE-RTN.
    OUT-EXIT.
        EXIT PROGRAM.
   *
   * * * * * * * * * * * *  * * * * * * * * * * * *
   *
    OPEN-FILE-RTN.
        OPEN OUTPUT MPLSTOUT.
        MOVE ' NO' TO OPN-FIL.
   *
   * * * * * * * * * * * *  * * * * * * * * * * * *
   *
    PROCESSING-LOOP.
        MOVE LINE-IMAGE TO WORK-RECORD.
        MOVE SPACES TO OUTPUT-RECORD.
        MOVE STATUS-CODE-WK TO STATUS-CODE.
        MOVE STATUS-DATE-WK TO STATUS-DATE.
        MOVE PRODUCT-TYPE-WK TO PRODUCT-TYPE.
        MOVE PRODUCTION-PLAN-WK TO PRODUCTION-PLAN.
        MOVE SHIPPING-DATE-WK TO SHIPPING-DATE.
        MOVE SHIPPING-ORDER-WK TO SHIPPING-ORDER.
        WRITE OUTPUT-RECORD.
   *
   * * * * * * * * * * * *  * * * * * * * * * * * *
   *
    CLOSE-FILE-RTN.
        CLOSE MPLSTOUT.
        MOVE ' NO' TO CLS-FIL.
/*
// DVC RES  // LBL $Y$LOD  // LFD LOD
// WORK1
// WORK2
// EXEC LNKEDT
// PARAM OUT=LOD
/$
     LOADM UOWNCD
     INCLUDE MYPROG
/*
/&
```

Figure 4–5. Sample of User Own Code (Part 3 of 3)

Explanation of your own code for MAPLST execution:

Lines 22 and 23 select and assign a MIRAM file called MPLSTOUT. This own code program writes data to the MPLSTOUT file during its processing of the spool file.

In lines 31–77, the format of the record sent to the MIRAM file is not the same as the RID 0 of the MAPPER 80 report. The output record does not need to match the RID 0 of the MAPPER 80 report when you use working storage to manipulate the MAPPER 80 report line to match the output record. Thus, you may select only the portions of the MAPPER 80 report you want.

Lines 78–88 define (in the linkage section) data transferred to MAPLST.

Line 79 defines information transferred to MAPLST in a single group item.

Line 87 adjusts the line length to fit the record size of the MAPPER 80 report.

Line 91 shows that you must use the linkage section group item on the procedure division USING clause.

While the counter is greater than 5 and less than 26, you perform the processing loop that writes a record to the MIRAM file after moving the MAPPER 80 report line from print image to WORK-RECORD (line 98).

Line 101 closes the MIRAM file when no more records are placed in the MIRAM output file.

Lines 102 and 103 use the EXIT PROGRAM statement to exit your own code.

Because this own code selects only the desired data fields, it places the line image in the WORKING-STORAGE section first. Then, it moves the desired fields to OUTPUT-RECORD and writes OUTPUT-RECORD to the MIRAM file.

Always use a WORKING-STORAGE section in your own code to eliminate the tab sets that are a part of the MAPPER 80 report line received from MAPLST. Always move the line image to WORKING-STORAGE first before removing tab sets (lines 113–122).

Lines 130–138 are the control statements that tell the linkage editor to create the own-code load module. Remember to place the own-code load module into the same load module library where MAPLST resides. In this example, the load module library is $Y$LOD.

4.5. MAPLST ERROR MESSAGES

Table 4–2 lists error messages issued by the MAPLST utility.

Table 4–2. MAPLST Error Messages

Error Message Number	Error Message Text
MPR00	MAPLST NORMAL TERMINATION
MPR11	PARAMETER ACCEPT ERROR
MPR12	ILLEGAL PARAMETER
MPR13	ILLEGAL MODE NUMBER
MPR14	ILLEGAL TYPE NUMBER
MPR15	ILLEGAL RID NUMBER
MPR16	ILLEGAL START POSITION
MPR17	ILLEGAL END POSITION
MPR20	MODE DOES NOT EXIST
MPR21	RID DOES NOT EXIST
MPR30	MAPPER DATA BASE ACCESS ERROR (nnnn)

5. MAPRAD Utility

5.1. WHAT THE MAPRAD UTILITY DOES

MAPRAD is a batch utility program that adds a report to the MAPPER 80 data base from a MIRAM file. Although MAPRAD can add any report to the data base, it is designed to add run reports.

Figure 5–1 shows a control flowchart of the MAPRAD utility.

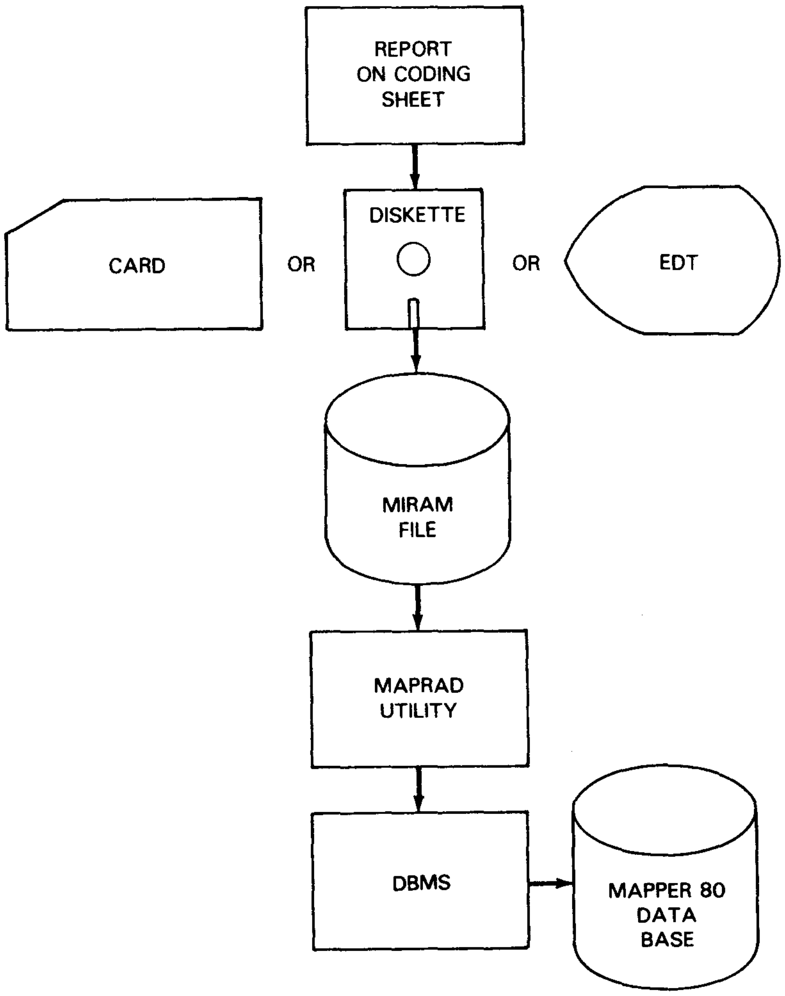


Figure 5–1. Control Flow for MAPRAD Utility Execution

The MAPRAD utility performs the following functions:

- Adds data from a MIRAM file to the MAPPER 80 data base as a report.
- Converts each line of the MIRAM file to a MAPPER 80 report line.
- Copies the headers on the MAPPER 80 report from the header lines of the report type RID 0. Also, inserts the tab characters in the proper location in the report, according to the RID 0 of the type.
- Replaces the specified report with a new report if it already exists in the MAPPER 80 data base.

5.2. MAPRAD CONTROL STATEMENTS

When you are ready to add reports to the MAPPER 80 data base from a MIRAM file, you execute the MAPRAD utility. MAPRAD control statements designate the user-id of the MIRAM source file; the mode number, type, and RID number where the report is stored; and a tab exchange character.

5.2.1. Naming the Creator of the Report (#UID)

The #UID control statement identifies the creator of the report being added to the MAPPER 80 data base.

Format:

```
#UID user-id
```

where:

`user-id`

Specifies the user-id (up to 12 characters) of the creator of a new MAPPER 80 report added to the MAPPER 80 data base.

When you omit this parameter, the default user-id, MAPPER, is used.

This control statement is not required when you use MAPRAD to modify or replace an already existing report on the MAPPER 80 data base.

5.2.2. Storing the Report (#ADD)

The #ADD control statement specifies the mode number, type, and optional RID number where the report is to be stored. It also optionally specifies a tab exchange character to be converted to tab settings on the MAPPER 80 report.

Format:

```
#ADD mmmt[rrr][c]
```

where:

mmm
: Is the mode number where the report will be stored. This parameter value must be 3 digits long with leading zeros, e.g., 4 is 004.

t
: Is the type where the report will be stored.

rrr
: Is the RID number of the report. When you omit this parameter, the lowest available RID number is assigned. This parameter value must be 3 digits long, and leading zeros are required.

c
: Is the tab exchange character used to convert any special character in the input data to a tab character in the MAPPER 80 report. Creating tab characters on input data is inconvenient, so using this parameter helps set up tab characters in the output report.

5.3. INPUT FILE REQUIREMENTS

Before executing the MAPRAD utility, the input file must have the following characteristics:

- The LFD name of the input file must be DISKIN. No other name can be used.
- When preparing the data input file using the general editor (EDT), the following parameters are required:

```
@w ,filename,vsn,RCFM=VAR,RCSZ=526,BFSZ=1024,INIT=Y
```

5.4. TAB LINE PROCESSING

When you add a data report (not run report) in the MAPPER 80 data base, you must specify the tab exchange character in the first column of every data line image in the input file. When the MAPRAD utility detects the tab exchange character in the first column of the line image, it converts the data line to the tab line and inserts the tab characters in the proper location in the line according to line 22 of the RID 0.

NOTE:

You must leave space between input data fields to accommodate tab characters; otherwise, you lose data characters within MAPPER 80 report fields.

5.5. MAPRAD EXECUTION

The following sample MAPRAD execution control stream includes the previously discussed MAPRAD control statements. Note that DBMS must be running before MAPRAD executes.

```
1.   // JOB MAPRAD,,12000
2.   // DVC 20   // LFD PRNTR
3.   // DVC 50   // VOL MAPPER
4.   // LBL MAPPER$REPORT   // LFD DISKIN
5.   // EXEC MAPRAD
6.   /$
7.   #UID MAPR
8.   #ADD 002B010?
9.   /*
10.  /&
11.  // FIN
```

The LBL statement (line 4) that specifies the input data file contains the image of the report. In the #UID control for MAPRAD (line 7), MAPR is the user-id name if report 10 doesn't exist in the MAPPER 80 data base. When report 10 does exist in the data base, the user-id name MAPR is ignored.

The #ADD control statement (line 8) indicates that the report will be created in mode 2, type B, and RID 10. The tab exchange character of a question mark (?) is converted to a tab character when MAPRAD executes.

5.6. MAPRAD ERROR MESSAGES

Table 5–1 lists error messages issued by the MAPRAD utility:

Table 5–1. MAPRAD Error Messages

Error Message Number	Error Message Text
MPR00	MAPRAD NORMAL TERMINATION MODE=mmm TYPE=t RID=rrr
MPR10	INPUT FILE OPEN ERROR
MPR12	ILLEGAL PARAMETER
MPR13	ILLEGAL MODE NUMBER
MPR14	ILLEGAL TYPE NUMBER
MPR15	ILLEGAL RID NUMBER
MPR20	MODE DOES NOT EXIST
MPR21	RID Ø DOES NOT EXIST
MPR30	MAPPER DATA BASE ACCESS ERROR
MPR40	MAPPER DATA BASE ACCESS ERROR

6. MAPDMS Utility

6.1. WHAT THE MAPDMS UTILITY DOES

MAPDMS is a utility that directly accesses the MAPPER 80 data base at a lower, more powerful level than the MAPLOD and MAPLST utilities. It requires more programming effort on the part of the user. As described earlier, you can write your own code that functions as a subroutine to the MAPLOD and MAPLST utilities.

With MAPDMS, however, the roles are reversed. You write the main program and MAPDMS operates as a set of subroutines to your program. This allows you to access the MAPPER 80 data base and execute complex operations without being familiar with the physical structure of the MAPPER 80 data base.

MAPDMS users can do the following types of processing:

- Prepare and enlarge reports
- Delete reports
- Search reports
- Update reports (including line addition, change, and deletion)

Figure 6–1 shows a control flowchart of the MAPDMS utility.

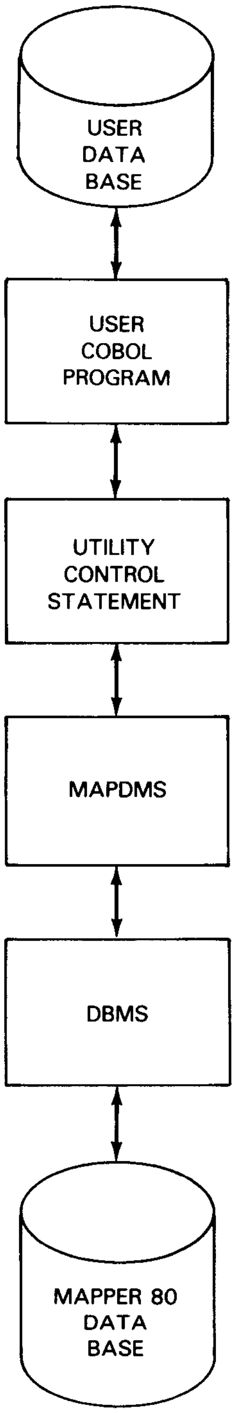


Figure 6–1. Control Flow for MAPDMS Utility Execution

6.2. MAPDMS FUNCTIONS

Your programs interface with MAPDMS utility using a number of functions that initiate or terminate MAPDMS and open, close, search, and update reports.

- Initiation/termination functions:
  - MP#SON initiates MAPDMS
  - MP#SOF terminates MAPDMS
- Open/close report functions:
  - MP#OPN opens a report for display or update
  - MP#CLS closes a report against display or update
- Inquiry/update functions:
  - MP#PLN positions the report
  - MP#GLN reads lines of the report
  - MP#ALN adds lines to a report
  - MP#MLN changes lines of a report
  - MP#DLN deletes lines of a report
  - MP#LLN loads lines of a report

6.2.1. CALL Function Statement Format

All CALL statements used by your programs have the following format:

```
CALL 'function-name' USING request-packet [data-area].
```

where:

`function-name`
: Identifies the function being called, e.g., MP#SON, MP#GLN, etc.

`request-packet`
: Specifies the name of the packet containing data to be transferred to and from MAPDMS. Your program prepares a packet for each type of data being transferred.

data-area

Names the data area used when transferring line records. It is necessary to specify this parameter when using the MP#GLN, MP#ALN, MP#MLN, and MP#LLN functions. Occasionally, it is specified for the MP#OPN function.

6.2.1.1. Contents of Request Packet

Each request packet is 23 bytes long and contains the following data:

RETURN-STATUS	(Unpacked format, 4 bytes) Gives the status of results returned from MAPDMS.
OPTION1	(Character format, 2 bytes) Sets function requests for detailed handling of function requests.
OPTION2	(Character format, 2 bytes) Sets any function requests not specified in OPTION1.
MODE-NO	(Unpacked format, 3 bytes) Specifies the mode number (000–999) where the processed report is stored. Only reports in even-numbered modes can be updated. Mode number is set when the function is requested.
TYPE-X	(Character format, 1 byte) Specifies the type letter (A–I) of the report being processed. Report type is set when the function is requested.
RID-NO	(Unpacked format, 3 bytes) Specifies the number of the report to be processed (001–999). RID number is set when the function is requested. This field can contain a RID from a MAPDMS result.
LINE-LENGTH	(Packed format, 2 bytes) Specifies the line length of the report being processed. This field is obtained as a result of MAPDMS execution.
C-LINE-NO	(Packed format, 3 bytes) Indicates the line number being processed (1–99999). This field is set when the function is requested. It can also be returned from a MAPDMS execution result.
NO-OF-LINES	(Packed format, 3 bytes) This is the number of lines in the report being processed, or the number of lines erased. This is set at the time of the request. It can also be a result returned from MAPDMS.

The following example shows how to code a request packet in your program. This request packet provides a place for your program to communicate with the MAPDMS utility.

```
WORKING-STORAGE SECTION.
Ø1  PACKET.
    Ø2 RETURN-STATUS    PIC  9(4) VALUE ZEROS.
    Ø2 OPTION1          PIC  XX   VALUE SPACES.
    Ø2 OPTION2          PIC  XX   VALUE SPACES.
    Ø2 MODE-NO          PIC  9(3) VALUE ZEROS.
    Ø2 TYPE-X           PIC  X    VALUE SPACE.
    Ø2 RID-NO           PIC  9(3) VALUE ZEROS.
    Ø2 LINE-LENGTH      PIC  9(3) COMP-3 VALUE ZEROS.
    Ø2 C-LINE-NO        PIC  9(5) COMP-3 VALUE ZEROS.
    Ø2 NO-OF-LINES      PIC  9(5) COMP-3 VALUE ZEROS.
```

6.2.1.2. Contents of Data Area

You need a data area when you use the MP#OPN, MP#GLN, MP#ALN, MP#MLN, or MP#LLN functions.

- MP#OPN function

  When a report is being created, expanded, or updated, you can transfer the following type of data to MAPDMS:

```
Ø1  DATA-AREA.
    Ø5  USER-ID          PIC X(12).
    Ø5  REPORT-PASSWORD  PIC X(6).
```

  When you specify this data, the user-id and report-password on line 1 of the resulting report are substituted for the specified contents. If you don't specify the data area when MP#OPN is called, blanks are set when creating a report and the original content remains when expanding or updating the report.

- MP#GLN, MP#ALN, MP#MLN, MP#LLN functions

  Line records are transferred in the following format:

```
Ø1  DATA-AREA.
    Ø5  LINE-RECORD-LENGTH     PIC 9(3)    COMP  VALUE ZEROS.
    Ø5  LINE-RECORD            PIC X(132).
```

The line record length must be set in binary before using the MP#GLN function. After MP#GLN is processed, the length of the transferred line record is set. When using other functions, you must set the lengths of line records transferred into MAPDMS in the range 80 to 132.

The line record is the report image of one line. When the line record of a tab line is transferred into MAPDMS, set a tab for the first column only (X'05'). All other tabs are set by MAPDMS.

6.2.2. Initiating and Terminating MAPDMS Utility (MP#SON and MP#SOF)

Your programs connect to the MAPDMS utility via the MP#SON function call. At that point, the MAPDMS utility initializes the necessary tables and buffers and begins initial processing of DMS.

The format for MP#SON is:

```
    CALL 'MP#SON' request-packet.
```

To establish the request-packet, make OPTION1 and OPTION2 blank.

Your programs disconnect from the MAPDMS utility with the MP#SOF function call.

The format for MP#SOF is:

```
    CALL 'MP#SOF' request-packet.
```

To establish the request-packet, specify on OPTION1 whether or not to validate an updated report to the present time. If OPTION1 is:

Blank	Validate the update and confirm the report update status. No updating actually occurs until the MP#SOF function is called with no OPTION1 parameter.
RB	Cancel the update and restore the report to its status before the update.

For examples of these functions, see 6.4.

6.2.3. Initiating a Report (MP#OPN)

Before referencing or updating report lines, your program calls the MP#OPN function and specifies the object report and mode.

MAPDMS processes one report at a time. Therefore, once the MP#OPN function is called, it cannot be called again for another report until the MP#CLS function is called.

The MP#OPN function call initiates use of a report.

The format for MP#OPN is:

```
CALL 'MP#OPN' request-packet.
```

To establish the request-packet:

`OPTION1`
Specifies the handling mode for the report:

`Blank`
Reference only is allowed. Only MP#PLN (line positioning) and MP#GLN (read) functions are possible (input mode).

`LD`
Creates a new report. Only the MP#LLN (write) function is possible. Your program writes only the actual data line. MAPDMS prepares headings. (See the FR parameter on OPTION2.)

`EX`
Extends the report. Only the MP#LLN function can be used. Lines can be added to the end of an existing report (extension mode).

`DT`
Deletes an entire report. The MP#CLS function must be called after this (delete mode).

`IO`
Allows changes to an existing report. This parameter allows all update functions except MP#LLN (update mode).

`OPTION2`
Specifies RID number of object report:

`Blank`
Makes the RID number in the packet the processing object. This can be specified in all handling modes. Except for the LD mode, the RID must exist at this time.

FR

Makes the object RID the lowest numbered open RID (within the specified type). This specification is valid only in LD mode. The RID number specified in the request packet is ignored. When this function ends, the object RID number is returned to the packet.

NX

Makes the object RID the next free (open) RID after the RID number specified in the packet. If RID numbers 004, 007, and 008 exist in one type, a specification of 005 in the packet causes 007 to become the processing object. At the end of this function call, the RID number is returned to the packet. This option can be specified with all modes.

MODE-NO

Supplies the number of mode containing the report being processed (values are 000 to 999). An odd mode number specifies read only.

TYPE-X

Specifies the type containing the report being processed (value A to I).

RID-NO

Is from 001 to 999, when specifying a blank for OPTION2. When specifying NX in OPTION2, specify from 000 to 998 as the RID number. Specify 000 when you want the lowest numbered RID within the type to be the report processed.

To set the data area, specify it when using the creation, extension, or update modes. The user-id and report password are set in the data area for later use.

MAPDMS updates the date and time in line 1 after creating or updating a report. It also updates the user-id and report password if this information is available in the data area.

MAPDMS returns the following data to the request packet:

RID-NO

Specifies the RID number of the object report.

LINE-LENGTH

Is the line length of the object RID set (80 to 132 columns).

C-LINE-NO

Is the current line number according to the handling mode:

Input mode: 1

Creation mode: usually 6 (number of line after header)

Extension mode: line number of the last line in the object report (END REPORT line)

Update mode: 1

NO-OF-LINES
: Is the number of lines in the object report (from 2 to 99999). This includes line 1 and the END REPORT line. Nothing is returned here when in creation mode.

The following examples show the use of options on the MP#OPN function call.

1. Referencing a specific report

```
MOVE BLANKS TO OPTION1, OPTION2.
CALL 'MP#OPN' request-packet.
.
.
.
CALL 'MP#CLS' request-packet.
```

2. Creating a new report for an open RID number

```
MOVE 'LD' TO OPTION1.
MOVE 'FR' TO OPTION2.
CALL 'MP#OPN' request-packet.
.
.
.
CALL 'MP#CLS' request-packet.
```

3. Creating a new report for a specific RID number

```
MOVE 'LD' TO OPTION1.
MOVE BLANKS TO OPTION2.
CALL 'MP#OPN' request-packet.
.
.
.
CALL 'MP#CLS' request-packet.
```

4. Updating multiple reports (one by one) from a certain RID number set in the packet. (The RID number set in the packet must immediately precede the first processed RID.)

```
MOVE 'IO' TO OPTION1.
MOVE 'NX' TO OPTION2.
CALL 'MP#OPN' request-packet.
.
.
.
CALL 'MP#CLS' request-packet.
```

6.2.4. Terminating a Report (MP#CLS)

When report processing is complete, your program specifies the end of report by calling the MP#CLS function.

The format for MP#CLS is:

```
CALL 'MP#CLS'  request-packet.
```

→ To set the request-packet, specify the MODE-NO, TYPE-X, and RID-NO of the report being processed.

MAPDMS returns the following data to the request-packet:

`NO-OF-LINES`
: Is the number of lines in report being processed. This value is not returned in delete mode.

For examples of this function, see 6.2.3.

6.2.5. Processing Report Line Reference Functions

You can call the following functions for processing line records:

`MP#PLN`	Positions a line
`MP#GLN`	Reads a line
`MP#ALN`	Adds a line
`MP#MLN`	Changes a line
`MP#DLN`	Deletes a line
`MP#LLN`	Loads a line

When using MP#GLN, MP#ALN, MP#MLN, and MP#LLN functions, one line at a time transfers into your program. Table 6–1 shows which handling modes (for MP#OPN) are used with each line processing function.

Table 6–1. Allowable Handling Modes for Processing Report Line Reference Functions

Processing Functions	**MP#OPN Handling Modes**				
	Input Mode	**Update Mode**	**Creation Mode**	**Extension Mode**	**Delete Mode**
	Blank	**IO**	**LD**	**EX**	**DT**
MP#PLN	X	X			
MP#GLN	X	X			
MP#ALN		X			
MP#MLN		X			
MP#DLN		X			X
MP#LLN			X	X	

6.2.5.1. Positioning a Line on a Report (MP#PLN)

The MP#PLN function positions the line at the specified line number. The positioned line becomes the current line. The current line becomes the object when a line function executes. This is the C-LINE-NO described in your program's working storage.

MAPDMS references and processes lines with the current line as the base of reference. The line numbers in a report signify the position of the lines within that report. Line 1 is the first line of a report and the other lines are numbered in ascending order (2, 3, 4, etc).

These line numbers are not found in the line record, and the determination of the position of a line from the beginning of the report is handled logically. When adding or deleting lines, the line numbers of all subsequent lines change.

The format for MP#PLN is:

```
CALL 'MP#PLN'  request-packet.
```

To set the request-packet:

`MODE-NO`
: Specifies the number of the mode containing the report being processed.

`TYPEX`
: Is the letter of the type containing the report being processed (value A to I).

`RID-NO`
: Is the RID number of the object report.

`C-LINE-NO`
: Is the line number where line positioning begins. Values are 1 to 99999. When specifying a line number larger than the last existing line, the current line number becomes the END REPORT line.

For examples of this function, see 6.2.5.2 and 6.2.5.3.

6.2.5.2. Reading the Next Line (MP#GLN)

The MP#GLN function reads the line following the current line and sets the line record in the specified data area. The line number of the transferred line record becomes the new current line.

If the program calls a MP#GLN function after requesting a MP#OPN, MP#PLN, or MP#DLN function, the current line does not change and the line record of the current line transfers into your program's data area.

The format for MP#GLN is:

```
    CALL 'MP#GLN'  request-packet  data-area.
```

To set the request-packet, specify the MODE-NO, TYPE-X, and RID-NO of the object report.

MAPDMS returns the following data to the request-packet:

`C-LINE-NO`
    Is the line number of the transferred line.

Examples:

1. Read one line at a time starting with the first line of the report:

```
        CALL 'MP#OPN' request-packet.                (Line 1 is the current line.)
        .
        .
        .
        CALL 'MP#GLN' request-packet data-area.      (Read from line 1 in order. This function
                                                     must execute once for every line read.)
```

2. Read one line at a time starting in the middle of the report.

```
        CALL 'MP#OPN' request-packet.
        CALL 'MP#PLN' request-packet.                (Positions the report where you want to
        .                                            begin to read. The current line number
        .                                            contains the desired line position.)
        .
        CALL 'MP#GLN' request-packet data-area.      (Read from the current line number
                                                     specified in the MP#PLN function. This
                                                     function executes once for every line
                                                     read.)
```

6.2.5.3. Adding a Line to a Report (MP#ALN)

The MP#ALN function adds the line record specified in the data area at the line following the current line number. The added line then becomes the current line. This function is invalid when the END REPORT line number is the current line.

The format for MP#ALN is:

```
    CALL 'MPALN'  request-packet  data-area.
```

To set the request-packet, specify the MODE-NO, TYPE-X, and RID-NO of the object report.

MAPDMS returns the following data to the request-packet:

`C-LINE-NO`
: Is set to the line number of the added line.

Examples:

1. Add several lines following a specific line:

```
CALL 'MP#PLN' request-packet.                  (The preceding line becomes the current line.
.                                              If the added lines must start at the tenth
.                                              line, position the current line at the ninth
.                                              line.)
CALL 'MP#ALN' request-packet data-area.        (Adds lines in order. Executes once for
                                               every line added.)
```

2. Add lines to the end of the report.

```
CALL 'MP#PLN' request-packet.                  (Positions current line number at the line
.                                              preceding the END REPORT line.)
.
.
CALL 'MP#ALN' request-packet data-area.        (Adds one line.)
```

NOTES:

1. *When a line is added, all subsequent line numbers are incremented by 1.*
2. *If the program calls the MP#GLN function immediately after the MP#ALN function, the line following the added line transfers into your data area.*
3. *Use the MP#LLN function when large numbers of lines are added.*

   When lines are added with MP#ALN, the MAPPER 80 line index is no longer complete. Thus, random referencing of many lines in the report takes longer (e.g., an operation specifying the line number or the binary find (BF) manual function).

6.2.5.4. Changing the Contents of a Line (MP#MLN)

The MP#MLN function changes the contents of a line.

The format for MP#MLN is:

```
CALL 'MP#MLN'  request-packet  data-area.
```

To set the request-packet, specify the MODE-NO, TYPEX, and RID-NO of the object report.

Examples:

1. Change the entire contents of a specific line.

`CALL 'MP#PLN' request-packet.`	(The line being changed becomes the current line.)
.	
.	
.	
`CALL 'MP#MLN' request-packet data-area.`	(Substitutes one line. Your program puts the new line record into the data area before calling the MP#MLN function.)

2. Change one portion of several lines.

`CALL 'MP#GLN' request-packet data-area.`	(Reads line records in order.)
.	
.	
.	
`CALL 'MP#MLN' request-packet data-area.`	(Replaces lines. Each of these functions executes once for every line.)

6.2.5.5. Deleting a Line from a Report (MP#DLN)

The MP#DLN function deletes a specified number of lines, beginning with the current line. This function is invalid if the current line is line 1 or the END REPORT line.

If the number of lines from the current line to the END REPORT line is less than the number of lines specified, lines are deleted up to the line immediately preceding the END REPORT line.

After this function call executes, the line after the deleted lines becomes the current line, but the line number of the current line does not change. Also, the line number of the lines following the deleted lines are decremented by the number of lines deleted.

The format for MP#DLN is:

```
CALL 'MP#DLN' request-packet.
```

To set the request-packet, specify the MODE-NO, TYPE-X, and RID-NO of the object report.

MAPDMS returns the following data to the request-packet:

`NO-OF-LINES`
: Is the actual number of lines deleted. If the END REPORT line is encountered while lines are being deleted, the number of lines returned is less than the number requested.

Examples:

1. Delete a specific line:

```
CALL 'MP#PLN' request-packet.            (The line being deleted becomes
.                                         the current line.)
.
.
CALL 'MP#DLN' request-packet.            (Deletes the line.)
```

2. Delete lines with identical conditions, while reading a report.

```
CALL 'MP#GLN' request-packet data-area.  (Reads lines in order. When the line
                                          should be deleted, the MP#DLN
.                                         function is performed and control
.                                         returns to MP#GLN to read another
.                                         line. Otherwise, the MP#GLN function
                                          is performed.)
CALL 'MP#DLN' request-packet.
```

6.2.5.6. Creating New Reports and Expanding Old Reports (MP#LLN)

The MP#LLN function creates a new report or expands existing reports.

The report headings are automatically prepared in the creation mode. The END REPORT line is automatically prepared by the termination function (MP#CLS).

MP#LLN is the only function allowed in creation and extension mode.

The format for MP#LLN is:

```
CALL 'MP#LLN'  request-packet  data-area.
```

To set the request-packet, specify MODE-NO, TYPE-X, and RID-NO of the object report.

MAPDMS returns the following data to the request-packet:

`C-LINE-NO`
: Is the number of the next line being written. C-LINE-NO equals the number of the line just written plus 1.

Examples:

1. Create a new report

```
CALL 'MP#OPN' request-packet.                (Opens creation mode. Prepares headers
.                                             automatically.)
.
.
CALL 'MP#LLN' request-packet data-area.      (Loads data lines in order after the
.                                             headers. Executes once for every line.)
.
.
CALL 'MP#CLS' request-packet.                (Writes END REPORT line, completing the
                                              report.)
```

2. Expand a report

```
CALL 'MP#OPN' request-packet.                (Opens the extension mode.)
.
.
.
CALL 'MP#LLN' request-packet data-area.      (Loads data lines in order from the
.                                             END REPORT line. Executes once for
.                                             every line.)
.
CALL 'MP#CLS' request-packet.                (Writes the END REPORT line completing
                                              the report.)
```

NOTES:

1. *When you specify an option other than FR on OPTION2 (report creation mode, blank RID 0), the contents of the original report are destroyed.*

2. *To immediately reference a newly created or expanded report, do the following:*

CALL 'MP#CLS' request-packet.	*Closes new or expanded report.*
CALL 'MP#OPN' request-packet.	*Opens report in the input or update mode.*

6.2.6. Checking Status after Function Execution

MAPDMS returns a status and detailed status code to your program after each execution of a function call your program issues. Your program must check the RETURN-STATUS field of the request-packet returned from MAPDMS.

The RETURN-STATUS field contains four characters in unpacked format. The leading two characters (status code) indicate the broad classification of errors. The final two characters indicate the details (detailed status code).

Table 6–2 shows the status settings for each function.

Table 6-2. MAPDMS Status Codes Returned to Request-Packet (Part 1 of 2)

Status Returned			MAPDMS Functions									
Status Codes	Detailed Status Codes	Status Code Meaning	MP#SON	MP#SOF	MP#OPN	MP#CLS	MP#PLN	MP#GLN	MP#ALN	MP#MLN	MP#DLN	MP#LLN
0 0	0 0	Normal termination	X	X	X	X	X	X	X	X	X	X
0 0	0 1	Normal termination; last line encountered					X	X			X	
0 0	1 2	Specified mode does not exist.			X							
0 0	1 3	No RID 0 in specified type.			X							
0 0	1 4	RID 0 edit code incorrect.			X							
0 0	1 5	Specified RID 0 does not exist.			X							
0 0	1 7	Number of lines > 99999							X			X
0 0	1 8	No blank RID			X							
0 0	2 5	Report is not being correctly formed.						X				
0 1	0 0	Lock cannot be cleared in reasonable time.	X		X							
0 1	0 1	Data base area is exhausted.			X	X		X				X
0 2	0 0	Function used in incorrect sequence.	X	X	X	X	X	X	X	X	X	X
0 2	0 1	Function not allowed in specific handling mode.					X	X	X	X	X	X
0 2	0 2	Specified report not open.				X	X	X	X	X	X	X
0 2	0 3	Function not allowed with current line.							X	X	X	
0 2	0 4	Option combination incorrect			X							
0 2	1 1	Mode number error			X							
0 2	1 2	Type letter error			X							
0 2	1 3	RID number error			X							
0 2	1 5	Current line number error					X					

Table 6–2. MAPDMS Status Codes Returned to Request-Packet (Part 2 of 2)

Status Returned			MAPDMS Functions									
Status Codes	Detailed Status Codes	Status Code Meaning	MP#SON	MP#SOF	MP#OPN	MP#CLS	MP#PLN	MP#GLN	MP#ALN	MP#MLN	MP#DLN	MP#LLN
0 2	1 6	Error in line number									X	
0 2	1 8	No data-area specification						X	X	X		X
0 2	1 9	Data-area contents incorrect						X	X	X		X
0 3	n n	Abnormality within MAPDMS	X	X	X	X	X	X	X	X	X	X

When the first positions in the RETURN-STATUS field contain 00, this indicates:

- A status where the operation terminated normally
- A status where the operation terminated and a warning about the data base status was issued

In either case, the operation can continue.

Table 6–3 describes functions with detailed status codes of 1 and status code of 0 (RETURN-STATUS of 0001).

Table 6–3. Functions with Detailed Status Codes of 1 (Status Code 0)

Function Call	Meaning
MP#GLN	END REPORT line (80 bytes) was transferred to the data-area.
MP#PLN MP#DLN	Line positioning or delete terminated normally.

Table 6–4 describes functions with detailed status codes other than 1 and status code of 0.

Table 6–4. Functions with Other Detailed Status Codes (Status Code 0)

Function Call	Detailed Status Codes	Meaning
MP#OPN	12, 13, 14, 15, 18	Invalid function. MP#CLS cannot be called for this report; an MP#OPN can be executed on another report.
MP#ALN MP#LLN	17	Invalid function; preceding function is secured; MAPDMS can be called.
MP#GLN	25	Report being incorrectly formed so MP#CLS is issued; another report can be processed.

When the first two positions in the RETURN-STATUS field contain 01, this indicates:

- Report creation or updating to that point must be completely restored to original status.
- MP#SON is invalid for MAPDMS.

Table 6–5 describes functions with detailed status codes of 0 and 1 and status code of 1.

When the first two positions in the RETURN-STATUS field contain 02, there is an error in your program. You must revise the program and rerun it. You must restore any report created or updated until that time to its original status. You must call the MP#SOF function.

When the first two positions in the RETURN-STATUS field contain 03, an abnormality has occurred in the MAPDMS utility or DBMS runs.

Table 6–5. Functions with Detailed Status Codes 0 and 1 (Status Code 1)

Function	Detailed Status Code	Meaning
MP#OPN MP#SON	0 0	Another user has a lock on the report; retry MP#SON and MP#OPN after a short wait.
MP#OPN	0 1	The full volume extent of the data base is inadequate; see coordinator. Delete reports or expand data base.

6.3. MAPDMS EXECUTION

MAPDMS cannot control multiple tasks as the online MAPPER 80 job can. Thus, you cannot run concurrent programs using the MAPDMS utility. Before executing MAPDMS, you must complete the following:

- Your program must be linked and edited independently (without MAPDMS). Store the load module of the program in the same load module library as the MAPDMS modules.

- To execute your program, use the following statement in the job control stream to assign the area for the dynamic loading module:

  ```
  // SFT DLOAD=(1,CØØØ)
  ```

  If your program calls SORT, specify the following statement:

  ```
  // SFT DLOAD=(2,1ØØØØ)
  ```

- Start the DBMS job.

Refer to 6.5 for an explanation of how to use the MAPDMS utility in your program.

6.4. MAPDMS ROLLBACK AND LOCKS

When an abnormality occurs during processing, or when your program specifies RB in a MP#SOF function call, this function returns the created or updated report to its original state. This function is common to the entire MAPPER 80 system. It relies on the quick-before-looks function of DMS.

No updates to old or newly created reports become permanent until your program calls the MP#SOF function:

```
CALL 'MP#SON' request-packet.
.
.
.
                                   (Update reports 2C, 5E, etc)
                                   (Create reports 2D, 1B, etc)
.
.
.
CALL 'MP#SOF' request-packet.      (All permanent changes made here.)
```

If any abnormalities occur within this block, the reports are restored to their original form. Calling the MP#SOF function with an RB option has the same effect.

We recommend that your program check the status code after every function call. When nonfatal errors occur (00nn), you can call MP#SOF with the RB option.

Examples:

1. Define multiple reports at the same time:

```
CALL 'MP#SON' request-packet.
CALL 'MP#OPN' request-packet.          Report A
.
.                                      (Report preparation, updating, or deletion)
.
CALL 'MP#CLS' request-packet.
CALL 'MP#OPN' request-packet.          Report B
.
.                                      (Report preparation, updating, or deletion)
.
CALL 'MP#CLS' request-packet.
CALL 'MP#SOF' request-packet.          (Reports A and B are defined at the same time.)
```

2. Restore the updated contents of multiple reports at the same time

```
CALL 'MP#SON' request-packet.
CALL 'MP#OPN' request-packet.          Report A
.
.                                      (Report preparation, updating, or deletion)
.
CALL 'MP#CLS' request-packet.
CALL 'MP#OPN' request-packet.          Report B
.
.                                      (Report preparation, updating, or deletion)
.
CALL 'MP#CLS' request-packet.
MOVE 'RB' TO OPTION1.
CALL 'MP#SOF'  RB.                     (Reports A and B are restored to original
                                       status at the same time.)
```

NOTE:

When an abnormality (other than status 00nn) occurs while report B is being processed, reports A and B are restored to original status at the same time as shown.

3. Define individual reports

```
CALL 'MP#SON' request-packet.
CALL 'MP#OPN' request-packet.      Report A
.
.                                  (Report preparation, updating, or deletion)
.
CALL 'MP#CLS' request-packet.
CALL 'MP#SOF' request-packet.      (Report A is defined and report A changes are
                                   made permanent in the data base.)
CALL 'MP#SON' request-packet.
CALL 'MP#OPN' request-packet.      Report B
.
.                                  (Report preparation, updating, or deletion)
CALL 'MP#CLS' request-packet.
CALL 'MP#SOF' request-packet.      (Report B is defined and report B changes are
                                   made permanent in the data base.)
```

In this example, if an abnormality occurs while report B is being processed, only report B is restored to original status. The changes to report A remain.

When creating large or multiple reports with the MAPDMS utility, assure that the DMS quick-before-looks file is large enough by using a secondary allocation. Making a report permanent (MP#SOF) or rolling it back frees the image in the DMS quick-before-looks file.

When executing the MAPDMS utility at the same time that you are interactively accessing the data base, you must be careful not to lock out large or critical parts of the data base. For a complete discussion of DMS locks, refer to data base management system (DMS) system support functions user guide/programmer reference, UP-8272 (current version).

Follow the same concurrency principles as for designing any DMS batch or online program. That is, divide the work into small, independent units (success-units) and commit or roll back the work done in that unit before going on to the next unit of work.

Without the success-unit concept, it is easy to design a program that locks out other users from accessing the data base. This becomes especially noticeable when others are using the data base interactively, the usual case with MAPPER 80. In general, DMS releases locks when your program calls the MP#SOF function or when MAPDMS returns an abnormal status (other than 00nn).

You cannot update a report while another user is referencing it or trying to update a report. The status error 0100 indicates that one user is trying to reference a report that another user is updating or referencing.

6.5. MAPDMS EXAMPLE

Figure 6–2 shows a sample COBOL program that interfaces with MAPPER 80 through the MAPDMS utility. Note that the group item, PACKET (in working-storage), is used to pass information from this COBOL program to MAPPER 80. The DATA-AREA contains the records that are written into the report.

The following options were used to compile the program:

```
AXNON=NO      AXREF=NO      CALLST=YES   CDMIO=YES     CMCSST=NO
CPYTXT=YES    DIAG=YES      DIAGWN=YES   PIPS=5        IMSCOD=NO
LIST=YES      LNKCON=YES    LSTREF=YES   LSTWTH=12Ø    MAP=NO
MXNON=NO      MXREF=NO      OBJLST=NO    OBJMOD=YES    PAGOVF=NO
SPRLST=NO     SPROUT=3      PROVER=NO    SYNCHK=NO     TRNADR=YES
TRUNC=YES
```

The compile stream used for the program example in Figure 6–1 follows:

```
// JOB MPDMST1
// DVC 2Ø    // LFD PRNTR
// DVC 5Ø    // VOL MAPPER
// LBL MAPPER$OBJ    // LFD MAPOBJ
// DVC 5Ø    // VOL MAPPER
// LBL MAPERR                // LFD ERRFIL
// WORK1
// WORK2
// WORK3
// EXEC COBL74
// PARAM OBJ=MAPOBJ
// PARAM CALLST=YES
// PARAM ERRFIL=MPDMERR/ERRFIL
/$
    (Place program here)
/*
/&
// FIN
```

NOTE:

The file MAPERR.MAPPER is used optionally by the error file processor and will contain the list of errors from the compile. This file must be allocated before program compilation by using the following allocate statement:

AL MI,FI=MAPERR,VSN=MAPPER,SIZ=2,INC=2 ←

Explanation:

This sample COBOL program reads a MIRAM data file (MPDTAIN) and converts it into a MAPPER 80 report. MAPDMS stores the report in mode 102, type B, and gives it the next available RID number (lines 80–81 and 88–89).

The CALL 'MP#SON' function call on line 75 signs you onto MAPPER 80 via the MAPDMS utility.

The CALL 'MP#OPN' function call on line 83 opens mode 102, type B, and prepares to create a new report.

The program then reads one record from file MPDTAIN and writes this record to the report via the MP#LLN function call.

This loop is repeated until eight records are written.

The CALL 'MP#CLS' function call on line 91 closes the mode and writes the report to the data base.

The CALL 'MP#SOF' function call on line 95 signs you off MAPPER 80 software.

The following job stream links and executes this COBOL program:

Link phase:

```
// JOB LNKMPDMS,,,1E000
// DVC 20    // LFD PRNTR
// DVC 50    // VOL MAPPER
// LBL MAPPER$OBJ    // LFD OBJ
// DVC 50    // VOL MAPPER
// LBL MAPPER$LOD    // LFD LOD
// WORK1
// WORK2
// EXEC LNKEDT
// PARAM OUT=LOD
/$
   LOADM  MPDMSL
   INCLUDE MAPDMS,OBJ
   INCLUDE MAPDMA,OBJ
   INCLUDE MAPDMC,OBJ
/*
```

Execute phase:

```
// DVC 2Ø   // LFD PRNTR
// DVC 5Ø   // VOL MAPPER
// LBL MAPPER$LOD   // LFD LOD
// DVC 5Ø   // VOL MAPPER
// LBL MPDTAIN   // LFD MAPIN
// SFT DLOAD=(15,7ØØØ)
// EXEC MPDMSL,LOD
/&
// FIN
```

NOTE:

MPDMS is the object module of the COBOL program. MAPDMA and MAPDMC must be included to run MAPDMS.

```
          IDENTIFICATION DIVISION.
          PROGRAM-ID.  MPDMS.
          AUTHOR       WAH AND SLA.
         *REMARKS.     THIS PROGRAM CALLS MAPDMS TO CREATE
         *             A MAPPER REPORT FROM A MIRAM FILE.
         *
         * * * * * * * * * * * *  * * * * * * * * * * * *
         *
          ENVIRONMENT DIVISION.
          CONFIGURATION SECTION.
          SOURCE-COMPUTER. UNIVAC-OS3.
          OBJECT-COMPUTER. UNIVAC-OS3.

          INPUT-OUTPUT SECTION.
          FILE-CONTROL.
              SELECT MPDTAIN
              ASSIGN TO DISK-MAPIN-F.
         *
         * * * * * * * * * * * *  * * * * * * * * * * * *
         *
          DATA DIVISION.
          FILE SECTION.
          FD  MPDTAIN
              LABEL RECORD IS STANDARD.
          01  INPUT-RECORD.
              05  STATUS-CODE              PIC X(2).
              05  STATUS-DATE              PIC 9(6).
              05  BY-IN                    PIC X(2).
              05  PRODUCT-TYPE             PIC X(9).
              05  SERIAL-NUMBER            PIC 9(6).
              05  PRODUCT-COST             PIC 9(6).
              05  ORDER-NUMBER             PIC 9(5).
              05  CUSTOMER-CODE            PIC X(4).
              05  PRODUCTION-PLAN          PIC 9(6).
              05  PRODUCTION-ACTUAL        PIC 9(6).
              05  SHIPPING-DATE            PIC 9(6).
              05  SHIPPING-ORDER           PIC X(5).
              05  SPECIAL-CODE             PIC X(3).
              05  REMARKS                  PIC X(9).
              05  FILLER                   PIC X(181).
          WORKING-STORAGE SECTION.
          01  PROGRAM-INDICATOR.
              05  REC                      PIC 9.
          01  PACKET.
              02  RETURN-STATUS             PIC 9(4) VALUE ZEROS.
              02  OPTION1                  PIC XX VALUE SPACES.
              02  OPTION2                  PIC XX VALUE SPACES.
              02  MODE-NO                  PIC 9(3) VALUE ZEROS.
              02  XTYPE                    PIC X VALUE SPACE.
              02  RID-NO                   PIC 9(3) VALUE ZEROS.
              02  LINE-LENGTH-P            PIC 9(3) COMP-3 VALUE ZERO.
              02  C-LINE-NO                PIC 9(5) COMP-3 VALUE ZEROS.
              02  NO-OF-LINES              PIC 9(5) COMP-3 VALUE ZEROS
          01   DATA-AREA.
               05   LINE-LENGTH    PIC 9(3) COMP VALUE ZEROS.
               05   LINE-RECORD-D.
                 10   LINE-TAB       PIC X VALUE X'05'.
                 10   LINE-RECORD    PIC X(79) VALUE SPACES.
          01   DATA-AREA-0.
               05   USER-ID  PIC X(12) VALUE 'JLOE
```

Figure 6–2. Sample COBOL Program to Interface MAPPER 80 via MAPDMS Utility (Part 1 of 3)

```
           05   PASSWRD PIC X(6)  VALUE SPACES.
      *
      * * * * * * * * * * * *   * * * * * * * * * * * *
      *
       PROCEDURE DIVISION.
      *
      * * * * * * * * * * * *   * * * * * * * * * * * *
      * THE MAIN-LINE PROGRAM CALLS IN THE LOWER-LEVEL
      * MODULES TO DO THE ACTUAL WORK OF THE PROGRAM.
      * THIS CENTRALIZES THE KEY LOGIC OF THE PROGRAM.
      * * * * * * * * * * * *   * * * * * * * * * * * *
      *
       MAIN-LINE.
           PERFORM HOUSE-KEEPING.
           CALL 'MP#SON' USING PACKET.
           IF RETURN-STATUS NOT EQUAL ZEROES THEN
              DISPLAY 'THIS IS RETURN STATUS SON:', RETURN-STATUS.
           MOVE 'LD' TO OPTION1.
           MOVE 'FR' TO OPTION2.
           MOVE '102' TO MODE-NO.
           MOVE 'B' TO XTYPE.
           DISPLAY REC , 'PACKET' , PACKET , 'DTA ARA' , DATA-AREA-O.
           CALL 'MP#OPN' USING PACKET , DATA-AREA-O.
           DISPLAY REC , 'PACKET' , PACKET , 'DTA ARA' , DATA-AREA-O.
           MOVE SPACES TO OPTION1 , OPTION2.
           IF RETURN-STATUS NOT EQUAL ZEROES THEN
              DISPLAY 'THIS IS RETURN STATUS OPN:', RETURN-STATUS
           PERFORM PROCESSING-LOOP VARYING REC FROM 1 BY 1 UNTIL
                       REC = 9.
           MOVE SPACES TO OPTION1 , OPTION2.
           CALL 'MP#CLS' USING PACKET
           IF RETURN-STATUS NOT EQUAL ZEROES THEN
              DISPLAY 'THIS IS RETURN STATUS CLS:', RETURN-STATUS.
           DISPLAY 'THIS IS THE RID-NO CREATED:' , RID-NO.
           CALL 'MP#SOF' USING PACKET.
           IF RETURN-STATUS NOT EQUAL ZEROES THEN
              DISPLAY 'THIS IS RETURN STATUS SOF:', RETURN-STATUS.
           PERFORM END-OF-JOB-RTN.
           STOP RUN.
      *
      * * * * * * * * * * * *   * * * * * * * * * * * *
      * THE HOUSE-KEEPING MODULE OPENS FILES AND DOES
      * OTHER INITIALIZATIONS.
      * * * * * * * * * * * *   * * * * * * * * * * * *
      *
       HOUSE-KEEPING.
           OPEN I-O MPDTAIN.
      *
      * * * * * * * * * * * *   * * * * * * * * * * * *
      * THE PROCESSING-LOOP DOES THE MAJOR WORK OF THE
      * PROGRAM.  IN THIS CASE, IT READS THE MIRAM INPUT
      * FILE.
      * * * * * * * * * * * *   * * * * * * * * * * * *
      *
       PROCESSING-LOOP.
           READ MPDTAIN
              AT END MOVE 9 TO REC.
           MOVE INPUT-RECORD TO LINE-RECORD.
           MOVE 80 TO LINE-LENGTH.
           CALL 'MP#LLN' USING PACKET DATA-AREA.
```

Figure 6–2. Sample COBOL Program to Interface MAPPER 80 via MAPDMS Utility (Part 2 of 3)

```
               IF RETURN-STATUS NOT EQUAL ZEROES THEN
                  DISPLAY 'THIS IS RETURN STATUS LLN:', RETURN-STATUS.
               DISPLAY REC.
               DISPLAY REC , 'PACKET' , PACKET , 'DATA AREA' , DATA-AREA.
           *
           * * * * * * * * * * * *  * * * * * * * * * * * *
           * THE END-OF-JOB-RTN TERMINATES THE PROGRAM AFTER
           * CLOSING FILES.
           * * * * * * * * * * * *  * * * * * * * * * * * *
           *
            END-OF-JOB-RTN.
               CLOSE MPDTAIN.
```

Figure 6–2. Sample COBOL Program to Interface MAPPER 80 via MAPDMS Utility (Part 3 of 3)

Appendix A. MAPPER 80 Error Messages

MAPPER 80 can issue error messages during an interactive session. These error messages appear on line 24 of the screen.

<ERR 090>(EXCEEDED REPORT LINE LIMIT)
Number of lines in the specified RID reached allowable limit (99999). Correct and retry.

<ERR 091>(NO FREE RID IN THIS TYPE)
Number of RIDs in this type reached allowable limit (999). Delete RIDs or go to another type or mode.

<ERR 092>(A LOCK–OUT HAS OCCURRED! PLEASE RE–TRY IN A MOMENT)
Specified RID is in use. Try again after a short wait.

<ERR 093>(NO SPACE IN MAPPER 80 DATA–BASE)
See MAPPER 80 coordinator or take job dump.

<ERR 094>(END REPORT WAS REACHED BEFORE FUNCTION COMPLETED)
Last line was reached during updating. All updating is done up to that point.

<ERR 095>(THERE ARE NO DATA LINES IN THIS REPORT)
Specified RID contains only headers, no data lines found. Correct and retry.

<ERR 098>(SYSTEM ERROR OCCURRED!)
See MAPPER 80 coordinator or take a job jump.

<ERR 100>(USER–ID IS INCORRECT)
Specified user-id not found at sign on. Possible causes:

1. Incorrect registration of the user-id.

2. Incorrect user-id.

Correct and retry.

<ERR 109>(USER–ID IS NOT REGISTERED)
Specified user-id not found at sign-on. Correct and retry.

<ERR 110>(INVALID DEPARTMENT NUMBER)
Specified department does not exist at sign-on. Correct and retry.

<ERR 111>(CANNOT USE DEPARTMENT NUMBER 0)
Correct and retry.

<ERR 115>(YOU MUST SIGN–ON TO AN EVEN MODE NUMBER)
Correct and retry.

<ERR 116>(INVALID TYPE SPECIFIED)
Incorrect type specified in FGEN function. Correct and retry.

<ERR 117>(INVALID LINE LENGTH)
Invalid line length specified in FGEN function. Correct and retry.

<ERR 118>(SPECIFIED TYPE DOESN'T EXIST)
Specified type not registered in GEN function. Type still in open status. Correct and retry.

<ERR 120>(INVALID USER PASSWORD)
Incorrect user password specified at sign-on. Correct and retry.

<ERR 130>(USER'S MODE DOESN'T EXIST)
Default MODE specified on user-id list does not exist. See MAPPER 80 coordinator.

<ERR 135>(RID 0 DOESN'T EXIST)
RID 0 doesn't exist for FGEN function. Correct and retry.

<ERR 136>(UNRECOVERABLE DBMS ERROR OCCURRED)
See MAPPER 80 coordinator or take a job dump.

<ERR 137>(INVALID FIELD NUMBER)
Correct and retry.

<ERR 138>(SPECIFIED MODE DOESN'T EXIST)
Correct and retry.

<ERR 139>(SPECIFIED MODE ALREADY EXISTS)
Correct and retry.

<ERR 140>(INVALID TYPE OR RID PARAMETER)
When -t specified for the RID and type, the type specified was different from the current displaying result type. Correct and retry.

<ERR 141>(INVALID TYPE PARAMETER)
Incorrect type specified. Correct and retry.

<ERR 145>(SYSTEM ERROR OCCURRED IN FGEN PROCESSING)
Unrecoverable error occurred during FGEN processing. See MAPPER 80 coordinator or take job dump.

<ERR 146>(INVALID EDIT CODE)
Correct and retry.

<ERR 147>(INVALID KANJI FIELD SIZE)
Length of Kanji field is not even. Correct and retry.

<ERR 148>(KANJI FIELD SIZE IS GREATER THAN ALLOWABLE MAX)
Kanji field size is greater than 240 characters. Correct and retry.

<ERR 149>(SPECIFIED TYPE DOESN'T EXIST)
Correct and retry.

<ERR 150>(CANNOT SPECIFY "–" FOR RID)
A hyphen (-) was specified for the RID with no report or result currently being displayed. Correct and retry.

<ERR 151>(INVALID RID CHARACTERS SPECIFIED)
Correct and retry.

<ERR 152>(RID PARAMETER WAS NOT SPECIFIED)
Correct and retry.

<ERR 158>(THE RID DOESN'T EXIST)
Specified RID does not exist.

<ERR 159>(THE RID IS ALREADY CATALOGUED)
Specified RID already exists.

<ERR 160>(INVALID MODE SPECIFIED FOR ISSUING REPORT)
First mode specified is incorrect. Correct and retry.

<ERR 165>(INVALID PARAMETER SPECIFIED)
Incorrect character entered in response to the FGEN screen. Valid responses are: Y, M, E, C, or N. Retry with correct response.

<ERR 166>(BAD CHARACTER ENTERED)
Incorrect character entered in response to the FGEN screen. Valid responses are: Y or E. Retry with correct response.

<ERR 167>(INVALID FORMAT NUMBER)
Valid formats range from 1 through 6. Correct and retry.

<ERR 168>(INPUT PARAMETER ERROR)
No parameter specified on FGEN screen. Correct and retry.

<ERR 169>(NUMBER OF FIELDS IS GREATER THAN ALLOWABLE MAX)
Maximum number of fields is greater than 75. Correct and retry.

<ERR 170>(REP,ADON: SECOND TYPE IS BAD; MA,MAU: FIRST TYPE IS BAD)
When executing REP or ADON function, the receiving (or second) specified type is invalid. When executing MA or MAU function, the issuing (or first) specified type is invalid. Correct and retry.

<ERR 175>(LINE LENGTH GREATER THAN ALLOWABLE MAX)
Line length specified in FGEN function is greater than line length specified in TGEN function. Correct and retry.

<ERR 176>(FIELD LENGTH IS GREATER THAN ALLOWABLE MAX)
Correct and retry.

<ERR 177>(REQUIRED PARAMETERS MISSING)
Field length or edit code is missing on FGEN screen. Correct and retry.

<ERR 178>(NUMERIC FIELD SIZE IS GREATER THAN 16 CHARACTERS)
Numeric fields cannot be greater than 16 characters. Correct and retry.

<ERR 179>(CANNOT CHANGE THE FORMAT NO. WHEN CONTINUING INPUT)
Enter the same format number when continuing format description on more than one FGEN screen. Correct and retry.

<ERR 180>(REP,ADON: SECOND RID IS BAD; MA,MAU: FIRST RID IS BAD)
When executing REP or ADON function, the receiving (or second) specified RID number is invalid. When executing MA or MAU function, the issuing (or first) specified RID number is invalid. Correct and retry.

<ERR 190>(INVALID LINE NUMBER)
The Line function on line 0 can only contain numbers. Correct and retry.

<ERR 199>(INVALID OPTIONS COMBINATIONS)
See MAPPER 80 manual functions user guide, UP-9735 (current version) for valid option combinations. Correct and retry.

<ERR 200>(INVALID NUMBER FOR RL)
The RL function on line 0 can have the following entries:

1. A plus sign (+)
2. A minus sign (–)
3. Any number
4. Either sign followed by any number.

Correct and retry.

<ERR 210>(INVALID FORMAT NUMBER)

The FMT function on line 0 can have the following entries:

1. Any number from 1 through 6
2. A blank (means basic format; 0)

Correct and retry.

<ERR 220>(INVALID FORMAT FOR ISSUING REPORT)

First format specified is invalid. Correct and retry.

<ERR 240>(INVALID NUMBER OF ADD LINES)

Correct and retry.

<ERR 241>(INVALID NUMBER OF DELETE LINES)

Correct and retry.

<ERR 250>(INVALID NUMBER OF COPIES FOR DUP LINES)

When using the DUPLICATE LINES function

```
] n x m
```

an invalid value was entered for n. Correct and retry.

<ERR 255>(PASSWORD MUST BE ENTERED)

Enter password and retry.

<ERR 256>(INVALID PASSWORD)

Correct and retry.

<ERR 260>(CANNOT USE PREDEFINED LINES)

RID 0 for this report type does not allow the m parameter in the following function:

```
] n + m
```

Correct and retry.

<ERR 261>(INVALID PREDEFINED LINE NUMBER)
M parameter in the following function is invalid:

```
] n + m
```

Correct and retry.

<ERR 265>(CANNOT UPDATE BECAUSE UPDATE RESULT DOESN'T EXIST)
@UPD or @DEL run function was entered when no result existed, or @RSM run function entered when no other function was active.

<ERR 266>(NO CONTINUATION FUNCTION IS CURRENTLY ACTIVE)
@RSM run function was entered when no continuation function was active.

<ERR 270>(INVALID LINE SPACING)
Valid options are 1 through 9. Correct and retry.

<ERR 318>(ONLY TYPE–N VARIABLES CAN BE INSERTED INTO TYPE–N VARIABLES)
If second parameter of an @INS run function is a variable of type-N, the first parameter must be type-N:

```
@ INS V1,V2
```

where:

`V1,V2`

Are both type-N.

Correct and retry.

<ERR 350>(INVALID LINE–LENGTH)
Incorrect line length specified in TGEN function.

Correct and retry.

<ERR 370>(INPUT BOTH MODE AND TYPE)
Either the MODE or TYPE was omitted. In the @BRK run function, both of these parameters must be specified or omitted. Correct and retry.

<ERR 371>(INVALID MODE NUMBER)
MODE specified by @BRK run function cannot be used in this run.

<ERR 372>(MODE DOES NOT EXIST)
Correct and retry.

<ERR 373>(TYPE DOES NOT EXIST)
Correct and retry.

<ERR 374>(CANNOT USE RESERVED WORDS ON OUTPUT LINE)
Reserved words cannot be described on the output line. Enter the reserved word into some variable. Then output the variable.

<ERR 375>(CANNOT WRITE DEFINED VARIABLE ON OUTPUT LINE)
Define variables before describing them on an output line.

<ERR 376>(DATA LENGTH IS TOO LONG)
Length of output line is greater than designated line length for the type. Shorten the output line and retry.

<ERR 377>(KANJI DATA IS NOT ALLOWED IN THIS FIELD)
Position of the type-N variable in the output line does not correspond to position of the edit-code-7 field. Correct and retry.

<ERR 378>(CANNOT USE RESERVED WORD)
Reserved words cannot be specified in @GTO, @RSR, @ESR, or @RUN run functions. Describe them by a variable in an @CHG run function. This variable can then be used in other run functions. Correct and retry.

<ERR 379>(LABEL EXCEEDS 3 CHARACTERS)
Labels of a @GTO or @RSR run function must be a number in the range of 1 to 999. Correct and retry.

<ERR 380>(INVALID LABEL)
Labels of a @GTO or @RSR run function must be a number in the range of 1 to 999. Correct and retry.

<ERR 381>(UNDEFINED LABEL)
Label specified by a @GTO or @RSR run function does not exist. Correct and retry.

<ERR 382>(YOU ARE TRYING TO GO TO A LINE BEYOND END OF REPORT)
Specified line number is beyond the last line of the run report. Correct and retry.

<ERR 383>(YOU ARE TRYING TO GO TO A LINE BEFORE HEADER)
Specified line number designates a header line or a line before the header. Correct and retry.

<ERR 384>(CANNOT USE DEFINED VARIABLE FOR LABEL)
Cannot specify a new variable description as a label. Define new variable before using it as a label. Correct and retry.

<ERR 385>(THE ONLY VARIABLE TYPE ALLOWED FOR LABEL ARE "A", "I")
Correct and retry.

<ERR 386>(INVALID PARAMETER)
An extra field is in the parameter list, or a field is missing in the parameter list. Correct and retry.

<ERR 387>(ONLY 1 PARAMETER IS ALLOWED IN 2ND FIELD)
Second field of the @CHG run function contains more than one parameter.

```
@CHG V1 23
```

Correct and retry.

<ERR 388>(INVALID PARAMETER IN 2ND FIELD)
Second field of the @CHG run function is not a variable (Vn) or INPUT$. Correct and retry.

<ERR 389>(INVALID USE OF VARIABLE IN 2nd FIELD)
Second field of the @CHG run function cannot be a partial variable:

Wrong example:

```
@CHG Vn(a-b) Vm
```

Correct and retry.

<ERR 390>(INCOMPLETE VARIABLE IN ISSUING DATA)
Issuing data of the @CHG run function cannot be a newly defined variable:

Wrong example:

```
@CHG Vn VnI4
```

Correct and retry.

<ERR 401>(INVALID PARAMETER)

Correct descriptions of the @RNM run function subfields are:

```
@RNM -n
@RNM,MODE,TYPE,RID -n
```

Correct and retry.

<ERR 402>(DUPLICATE RESULT NUMBER)

The RID and the –n parameters have the same value. Change one of these parameters and retry command.

<ERR 403>(CANNOT USE VARIABLE FOR PARAMETER)

Correct and retry.

<ERR 404>(THE ONLY RENAMED RIDS ALLOWED ARE –0,–1,–2,–3,–4)

Correct and retry.

<ERR 405>(INVALID SFS SCREEN–NAME)

The screen-name should be:

Uxxx

where:

xxx
: Is a number in the range of 1 to 999.

Correct and retry.

<ERR 407>(DATA SIZE EXCEEDS 1920 CHARACTERS)

Total number of characters in input and output data exceeds 1920. Correct and retry.

<ERR 408>(USE VARIABLE FOR PARAMETER)

Parameters that receive input data in the @SFS run function must be variables. Correct and retry.

<ERR 409>(SFS SCREEN–NAME DOESN'T EXIST)

Correct and retry.

<ERR 391>(INVALID USE OF TYPE-N VARIABLE)

If the receiving variable of a @CHG run function is type-N, the issuing variable must also be type N. Both variables must be the same length. Correct and retry.

<ERR 392>(CANNOT UPDATE RID 0)

Line update functions cannot be performed on RID 0.

<ERR 393>(LOCK MUST BE DONE BEFORE YOU CAN UPDATE REPORT)

Report lock must be done before performing the following run functions:

```
@LN+, @LN-, @LNX, @WRL.
```

Lock the report (using @LOK) before updating or writing it.

<ERR 394>(YOU LOCKED DIFFERENT REPORT)

You did not lock the report you tried to update. Lock the report (using @LOK), then execute the update or write command.

<ERR 395>(INVALID NUMBER OF ADD LINE)

You tried to add lines beyond the range limit. All lines up to the limit were added.

<ERR 396>(INVALID PREDEFINED LINE NUMBER)

Predefined line number of the @LN+ run function must be a space or a number from 0 to 9. Correct and retry.

<ERR 397>(CANNOT UPDATE LINE–1)

Line number of a @LN+ run function cannot be 1. Line 1 cannot be updated.

<ERR 398>(CANNOT UPDATE BEYOND END LINE)

Specified line number of the @LN+ run function is beyond the END REPORT line. Lines are added up to this point.

<ERR 399>(CANNOT ADD LINE INTO HEADER)

The specified line number of the @LN+ run function is in the header lines. Correct and retry.

<ERR 400>(INVALID LABEL RUN WILL LOOP)

The specified line number of the @RSR run function designates the line on which the @RSR run function is described. Correct and retry.

<ERR 420>(INVALID EDIT-CODE IN RID 0)
Invalid edit-codes are in RID 0. Correct and retry.

<ERR 421>(SYSTEM ERROR OCCURRED (WRL))
System error occurs during internal processing of Kanji characters. See MAPPER 80 coordinator or take job dump.

<ERR 422>(CANNOT SPECIFY RESULT IN THIS FUNCTION)
Correct and retry.

<ERR 423>(INVALID USER–ID)
User-id not registered to execute the specified run.

<ERR 424>(CANNOT EXECUTE THE RUN ON THIS WORKSTATION)
Workstation not registered to execute the specified run.

<ERR 425>(THERE IS INCORRECT MODE IN RUN–LIST)
Incorrect registration of accessible modes in the run list.

<ERR 426>(NUMBER OF LINES OF REPORT EXCEEDS 99999)
Line duplication carried out until END REPORT reached.

<ERR 427>(INVALID RUN REPORT IN RUN–LIST)
Registration of mode-type-RID in the run-list is incorrect. Correct and retry.

<ERR 428>(ISSSUING FIELD LENGTH DOES NOT MATCH RECEIVING VARIABLE)
Correct and retry.

<ERR 450>(INVALID MARK (KANJI))
Kanji characters cannot be entered in this mask. Correct and retry.

<ERR 451>(UNRECOVERABLE ERROR OCCURRED IN DATA MANAGEMENT)
Error occurred in data management request to the workstation file. See MAPPER 80 coordinator or take job dump.

<ERR 410>(OUTPUT DATA DOES NOT MATCH SFS SCREEN ATTRIBUTE)
Size or character type of output data specified in the @SFS run function does not match the cataloged screen format. Correct and retry.

<ERR 411>(INVALID INPUT DATA ON SFS SCREEN)
Correct and retry.

<ERR 412>(INVALID LINE NUMBER)
Starting line number of the @RDL, @RDC, and @RLN run functions should not be zero. Correct and retry.

<ERR 413>(CANNOT SPECIFY "RLN" BEFORE "RDL")
Execute @RLN function after the @RDL function. Correct and retry.

<ERR 414>(CANNOT USE "INPUT$")
Reserved word INPUT$ cannot be used as issuing data in the @CHG run function.

<ERR 415>(CANNOT SPECIFY 0 FOR NO. OF LINE)
Number of lines processed cannot be zero in the @RDC run function. Correct and retry.

<ERR 416>(SPECIFY PARTIAL VARIABLE IN RECEIVING AREA)
Receiving data must be a partial variable [Vn(a-b)] in the @INS run function. Correct and retry.

<ERR 417>(DIFFERENT DATA SIZE BETWEEN TWO FIELDS)
Data sizes of the issuing and receiving fields are not equal in the @INS run function. Correct and retry.

<ERR 418>(MUST BE VARIABLE OF TYPE N)
Insert only Kanji data into type-N variables. Correct and retry.

<ERR 419>(CANNOT INSERT VARIABLE OF TYPE N)
A type-N variable can't be inserted into a non-type-N variable. Correct and retry.

<ERR 460>(INVALID FUNCTION NAME)
Specified function name is invalid. Possible causes are:

1. The first character of the function name is not a letter (A through Z) or the caret (∧) character.

2. The function name is not cataloged.

Correct and retry.

<ERR 461>(FUNCTION NAME IS TOO LONG)
Length of the specified function name is greater than 4 characters. Correct and retry.

<ERR 462>(FUNCTION NOT ALLOWED)
Workstation will not accept the specified function. Try a different function or enter the caret character (∧), transmit, and then try a different function.

<ERR 463>(INVALID POSITION OF PARAMETER FOR LINE–0)
SOE position of the line 0 function (LINE,FMT,etc) is invalid. Correct and retry.

<ERR 464>(INVALID PARAMETER FOR LINE MODIFICATION FUNCTIONS)
Line update function parameter is incorrect. Correct and retry.

<ERR 465>(CANNOT USE 'CC' OR 'FLD' FUNCTION WHEN LINES ARE HELD)
Correct and retry.

<ERR 470>(INPUT PARAMETER LENGTH SHOULD BE WITHIN 30 CHARACTERS)
When using the fast access method for function input, the length of the input parameter string cannot be greater than 30 characters. Correct and retry.

<ERR 471>(PARAMETER LENGTH IS TOO LONG)
Correct and retry.

<ERR 472>(INVALID CHARACTER STRING)
Input parameter contains an invalid character. Correct and retry.

<ERR 473>(NO NUMERIC CHARACTERS FOR NUMERIC PARAMETER)
A numeric parameter does not contain numeric characters. Correct and retry.

<ERR 474>(TYPE SHOULD BE 'A'–'I')
Correct and retry.

<ERR 475>(NO PARAMETER FOR REQUIRED FIELD)
Correct and retry.

<ERR 476>(INVALID REQUEST FOR HOLD CHARACTER)
Number of characters specified by SHFT function must be greater than the number specified in HLD CHR. Correct and retry.

<ERR 477>(INVALID REQUEST FOR HOLD LINE)
Number of lines specified by the RL function must be greater than the number specified in HLD LN. Correct and retry.

<ERR 478>(NO NEGATIVE NOS ALLOWED IN HLD LN FIELD)
The HLD LN field must contain only positive numbers (or zero). Correct and retry.

<ERR 479>(INVALID FORMAT NUMBER)
FMT characters must be numeric (0 through 6) or blank (implies format 0). Correct and retry.

<ERR 480>(CANNOT MODIFY HELD LINES)
Line update functions cannot be performed on held lines.

<ERR 481>(CANNOT MODIFY LINE–1)
Line update functions cannot be performed on line 1.

<ERR 482>(CANNOT MODIFY BEYOND THE LAST LINE)
Line update functions cannot be performed in the area after the END REPORT line. Add lines, then use line update.

<ERR 483>(INVALID SOE POSITION)
The SOE should be on the top line of the screen. Move SOE and retry.

<ERR 484>(CANNOT INCLUDE LAST LINE IN HELD LINES)
END REPORT line cannot be included in the HLD LN function.

<ERR 485>(INVALID PARAMETER FOR MASK)
Parameter input to the mask is incomplete. Correct and retry.

<ERR 486>(CANNOT UPDATE AREA BEYOND THE LINE LENGTH)
Area is updated beyond the designated line length of the report, and the report cannot be updated beyond the length of the report.

<ERR 487>(CANNOT MODIFY HEADER LINES)
The only line update function that can be performed on header lines is the SOE update function. All others are invalid.

<ERR 488>(NO TAB CHARACTER WAS ENTERED IN A DESIGNATED TAB COLUMN)
When performing an update line function, enter a tab code character in a tab designated column. Correct and retry.

<ERR 489>(NO NUMERIC CHARACTER ENTERED IN NUMERIC FIELD)
Correct and retry.

<ERR 490>(A BLANK WAS ENTERED TO A FIELD WHICH CANNOT BE BLANK)
During a line update function, a blank was entered to a nonblank field (edit-code-5). Correct and retry.

<ERR 491>(NO ALPHA CHARACTER ENTERED IN DESIGNATED ALPHA FIELD)
Designated alphabetic parameter has nonalphabetic characters. Correct and retry.

<ERR 492>(CANNOT INPUT KANJI ON OPTION LINE)
Kanji characters can't be entered on the option input line of the mask. Correct and retry.

<ERR 493>(CANNOT ENTER KANJI CHARACTERS ON LINE 0)
Correct and retry.

<ERR 494>(CANNOT UPDATE RIGHT COLUMN OF KANJI CHARACTER)
The right byte of the 2-byte Kanji character can't be changed.

<ERR 495>(CANNOT UPDATE LEFT COLUMN OF KANJI CHARACTER)
The left byte of the 2-byte Kanji character can't be changed.

<ERR 496>(INVALID KANJI CHARACTER)
Incorrect update to a Kanji character. Correct and retry.

<ERR 497>(KANJI DATA IS NOT ALLOWED ON THIS LINE)
During a line update function, Kanji characters were entered on a line with no edit-code-7 fields. Correct and retry.

<ERR 498>(CANNOT INPUT KANJI)
During a line update function, Kanji characters were entered in a non-Kanji field. Correct and retry.

<ERR 499>(INPUT IS IGNORED. PLEASE RE-TRY)
Data format of the input is incorrect. Correct and retry.

<ERR 500>(CANNOT USE COORDINATOR'S FUNCTION)
Only a MAPPER 80 coordinator can use coordinator functions.

<ERR 501>(INVALID FUNCTION REQUEST)
Update functions cannot be performed on reports of odd modes.

<ERR 502>(CANNOT DESIGNATE RID 0 IN THE FUNCTION)
Function cannot process the RID 0 report.

<ERR 503>(CANNOT DESIGNATE RID IN THE FUNCTION)
Function cannot process the RID 0 report.

<ERR 504>(CANNOT DESIGNATE RID 0 IN THE FUNCTION)
Function cannot process RID 0 as the second RID specification. Correct and retry.

<ERR 520>(THERE IS NO PREVIOUS RESULT)
PRED function executed when no previous result existed.

<ERR 548>(CANNOT PROCESS MULTIPLE RIDs)
SU function does not process multiple RIDs. It must operate on one RID at a time.

<ERR 549>(TOO MANY SUBFIELDS IN CHARACTER-SPECIFYING FIELD)
In the @RUN run function, the number of subfields in the character-specifying field is greater than the number of parameters in the parameter specifying field. Correct and retry.

<ERR 550>(TOO MANY NESTED SUBROUTINES)
@RSR run function can be nested to 10 levels only.

<ERR 551>(INVALID ARITHMETIC DATA)
Data to be calculated is not in the correct numerical format. Correct and retry.

<ERR 552>(INVALID USE OF ESR FUNCTION)
An @ESR run function can only be the last statement of a subroutine. Every @ESR function must be paired with a @RSR function. Correct and retry.

<ERR 553>(CANNOT USE DEFINED VARIABLE AS A BIAS)
In the @ESR run function, a newly defined variable cannot be used as a bias. Define variable before using and retry.

<ERR 554>(INVALID VARIABLE TYPE FOR BIAS)
Only a variable of type A or I can be the bias of an @ESR run function. Correct and retry.

<ERR 555>(INVALID BIAS)
Possible causes of this error:

1. Bias is not in the correct numerical format.
2. Format of the bias is too long.

Correct and retry.

<ERR 556>(BIAS EXCEEDS END LINE)
Line specified by the bias is not in range of the run report lines. Correct and retry.

<ERR 505>(PLEASE DISPLAY AGAIN: THIS REPORT WAS UPDATED!)
While displaying this report, another user updated it. Redisplay the report to see updated version.

<ERR 510>(INVALID DELETION OF * IN MASK)
Correct and retry.

<ERR 511>(BAD KEY FIELD: TAB CODE WAS SPECIFIED AS KEY)
Tab code column specified as the key-field in the SORT function mask. Correct and retry.

<ERR 512>(INVALID SORTING PRIORITY)
Incorrect sorting levels specified in the SORT function mask. Correct and retry.

<ERR 513>(MISSING KEY FIELD)
No parameter specified as the SORT key. Correct and retry.

<ERR 514>(INVALID SORTING PRIORITY)
Priority numbers of the SORT keys fields are not consecutive. Correct and retry.

<ERR 515>(MAX LENGTH OF SORT FIELD EXCEEDED)
Total length of SORT field exceeds limit of 36 characters. Correct and retry.

<ERR 516>(DUPLICATE SORTING PRIORITY)
Two SORT fields have the same priority. They must be different. Correct and retry.

<ERR 517>(INVALID SORTING SEQUENCE)
Errors are in the priority numbers of the SORT fields. Correct and retry.

<ERR 518>(CANNOT SPECIFY KANJI DATA AS NUMERIC DATA)
Numeric sorting cannot be specified for Kanji character fields. Correct and retry.

<ERR 519>(INVALID LINE TYPE)
An error occurred in the line type specification of the SORT mask. Correct and retry.

<ERR 557>(CANNOT USE DEFINED VARIABLE FOR RUN–NAME)
A newly defined variable cannot be specified as the run name in the @RUN run function. Correct and retry.

<ERR 558>(INVALID VARIABLE TYPE FOR RUN NAME)
A variable of type N, F, or I cannot be specified as a run name in the @RUN run function. Correct and retry.

<ERR 559>(LENGTH OF RUN NAME EXCEEDS 12 CHARACTERS)
Length of the run name must be 12 characters or less. Correct and retry.

<ERR 560>(RUN COMMAND DATA TRANSFER EXCEEDS LIMIT OF 40)
Number of variables sent to a second RUN is limited to 40. Correct and retry.

<ERR 561>(CANNOT USE RESERVED WORD)
Reserved words cannot be sent to a second run in a data transfer. Set a variable to the value of the reserved word, then send the variable. Correct and retry.

<ERR 562>(CANNOT USE DEFINED VARIABLE AS DATA)
Newly defined variables cannot be sent to a second run in a data transfer. Define the variable before sending it in a @RUN run function. Correct and retry.

<ERR 563>(DATA EXCEEDS 16 BYTES)
Length of inserting data exceeds 16 characters. Correct and retry.

<ERR 564>(TOTAL DATA SIZE EXCEEDS 320 BYTES)
Total length of sending data exceeds 320 characters. Correct and retry.

<ERR 565>(INVALID DIVISION BY ZERO)
Correct and retry.

<ERR 566>(RECEIVING PARAMETER FROM INPUT$ IS NOT VARIABLE)
Parameter that receives data from INPUT$ must be a variable. Correct and retry.

<ERR 567>(TOO MANY PARAMETERS)
Errors occur in the parameter specification of the @IF run function. Correct and retry.

<ERR 568>(INVALID VARIABLE COMPARISON)
Variable compared in the @IF run function is incorrect. Correct and retry.

<ERR 569>(COMPARISON DATA EXCEEDS 64 CHARACTER)
Length of the comparison variable is greater than 64 characters. Correct and retry.

<ERR 570>(INVALID ARITHMETIC OPERATOR)
Comparison expression in the @IF run function is incorrect. Correct and retry.

<ERR 571>(INVALID EXPRESSION FOR OR CONDITION)
Incorrect OR comparison expression occurs in the @IF run function. Correct and retry.

<ERR 572>(INVALID EXPRESSION (">" AND "<")
There are errors in the range condition of the @IF run function. Correct and retry.

<ERR 573>(INVALID USE OF VARIABLE TYPE–N)
Type-N variables can be compared only with other type-n variables. Correct and retry.

<ERR 574>(CHARACTER TYPE DOES NOT MATCH DESIGNATED EDIT–CODE)
Error in output line. Characters in the tab line do not match the edit-code specifications. Correct and retry.

<ERR 575>(DESIGNATED RUN IS WORKING NOW)
Run name specified in the @RUN run function cannot be the same as run where the command appears (i.e., a @RUN function cannot call itself).

<ERR 576>(INVALID RUN NAME)
Run name specified in the @RUN run function is invalid. Correct and retry.

<ERR 577>(CANNOT USE VARIABLE AND RESERVED WORD IN OPTION FIELD)
Correct @SUB run function and retry.

<ERR 578>(INVALID OPTIONS)
Incorrect options occur in the @SUB run function. Correct and retry.

<ERR 579>(INVALID OPTIONS COMBINATION)
Options are incorrectly combined in the @SUB run function. Correct and retry.

<ERR 580>(NO. OF VARIABLES EXCEEDS 40)
Number of variables receiving data from INPUT$ is greater than 40. Correct and retry.

<ERR 581>(CANNOT HAVE CONTIGUOUS EDIT-CODE-0s)
Two tab columns can't be designated contiguously.. Correct and retry.

<ERR 582>(SYSTEM ERROR: ADDRESS OF SFLE IS 0)
Error occurred in the internal processing of the run. See MAPPER 80 coordinator or take a job dump.

<ERR 583>(CANNOT EXCEED 3 CHARACTERS FOR SPECIFYING THE CHARACTER FIELD.)
Correct and retry.

<ERR 584>(INVALID DESCRIPTION IN CHARACTER FIELD)
Correct and retry.

<ERR 585>(INVALID FIELD NUMBER (Fxxx))
Character field description Fnnn is incorrect. Correct and retry.

<ERR 586>(DESIGNATED FIELD EXCEEDS LINE LENGTH)
Correct and retry.

<ERR 587>(OVERLAPPING FIELD SPECIFICATION)
Correct and retry.

<ERR 588>(LINE LIMIT EXCEEDED)
Too many slashes are in parameter field. Correct and retry.

<ERR 589>(REQUIRED FIELD MISSING)
Correct and retry.

<ERR 590>(LINE TYPE MUST BE 1 BYTE)
Line type must be one character in length. Correct and retry.

<ERR 591>(MAXIMUM PARAMETER EXCEEDED)
Number of data items in the parameter field exceeds limit. Correct and retry.

<ERR 592>(LENGTH OF CHARACTER FIELD IS DIFFERENT FROM PARAMETER FIELD)
The length specified in the parameter field, and number of characters in the character field must be the same. Correct and retry.

<ERR 593>(CANNOT COMBINE EDIT-CODE "7" FIELD WITH ANOTHER FIELD)
Correct and retry.

<ERR 594>(INVALID EDIT–CODE "7" FIELD)
Correct and retry.

<ERR 595>(EDIT-CODE AND VARIABLE TYPE DO NOT MATCH)
In the @RDL, @RLN, and @RDC run functions, variable type must match corresponding edit-code for that field. Correct and retry.

<ERR 596>(EDIT CODE AND DATA TYPE DO NOT MATCH)
Data type of the character does not match corresponding edit-code for that field. Correct and retry.

<ERR 597>(INVALID LEADING BLANK IN FIELD SPECIFICATION)
Incorrect field specification entered. Correct and retry.

<ERR 598>(INVALID SPACES IN CHARACTER FIELD SPECIFICATION)
Spaces cannot be specified in the character field.

<ERR 599>(INVALID DELIMITER ("–")
A hyphen (-) can't occur in the parameter field. Only a slash (/) and a comma (,) are permitted. Correct and retry.

<ERR 600>(SYSTEM ERROR IN FC)
Error occurred in MPDP's FC function. See MAPPER 80 coordinator or take a job dump.

<ERR 601>(SYSTEM ERROR IN FNC)

Error occurred in interface between FPC and FPR. An unregistered function name was sent to FPR from FPC. Correct and retry.

<ERR 605>(SYSTEM ERROR: UNREGISTERED OR UNCATALOGUED MODULE WAS CALLED)

System error was caused:

1. Unregistered program module called
2. Uncataloged shared code module of run (RUNSB1) was called. See MAPPER 80 coordinator or take a job dump.

<ERR 606>(SYSTEM ERROR IN LD)

LOADR MACRO was requested. See MAPPER 80 coordinator or take a job dump.

<ERR 607>(CAN'T GET MODULE LOAD AREA.)

Wait a few minutes. Then retry.

<ERR 610>(SYSTEM ERROR IN RB)

Error occurred in the MPDP-RB function. See MAPPER 80 coordinator or take a job dump.

<ERR 630>(CONTIGUOUS EDIT–CODE 0 IS INVALID)

Two tab columns can't be designated contiguously. Correct and retry.

<ERR 631>(FIELD SIZE OF EDIT–CODE "7" SHOULD BE EVEN)

All Kanji character fields must be even in length. Correct length and retry.

<ERR 640>(SYSTEM ERROR IN SORT)

Error occurred when requesting FPRSOW module from the FPRSRT module. See MAPPER 80 coordinator or take a job dump.

<ERR 641>(SYSTEM ERROR IN SORT)

Error occurred in MPDP sort request. See MAPPER 80 coordinator or take a job dump.

<ERR 650>(SPECIFIED MODE IS NOT ALLOWED TO THIS USER–ID)

Mode number specified in the M function is not accessible to this user-id.

<ERR 651>(INVALID MODE PASSWORD)

Mode password specified in the M function is incorrect.

<ERR 652>(MODE NUMBER OUT OF RANGE)
Mode number exceeds range 000 through 999. Correct and retry.

<ERR 653>(SYSTEM ERROR: TYPE SEQUENCE IS NOT B – I)
Type sequence in the data base does not run from B to I sequentially. See MAPPER 80 coordinator.

<ERR 685>(SPECIFIED REPORT MUST EQUAL DISPLAYING REPORT)
When using the DR function, the specified RID must correspond to currently displaying RID. First display the RID, then delete it.

<ERR 686>(THIS USER–ID CANNOT DELETE THIS REPORT)
To delete a RID, user-id must match the user-id of the RID.

<ERR 687>(THE REPORT ALREADY EXISTS)
You cannot add a report that already exists.

<ERR 688>(THERE IS NO FREE RID IN THIS TYPE)
No free RIDs are available. If possible, delete RIDs to make space for new ones.

<ERR 690>(REPLACED REPORT DOES NOT EXIST)
Second RID specified does not exist.

<ERR 691>(THIS USER–ID CANNOT REPLACE REPORT)
User-id does not match the second report (the replaced report). You cannot overwrite this RID.

<ERR 692>(THESE REPORTS MUST HAVE THE SAME LINE LENGTHS)
Line lengths of the replacing and replaced reports must be equal.

<ERR 693>(THERE IS NO FREE RID)
No free RIDs exist under the specified type. Delete some RIDs in order to add more to this type.

<ERR 695>(APPEND RID DOES NOT EXIST)
Specified adding RID (the second RID) does not exist.

<ERR 696>(THESE REPORTS MUST HAVE THE SAME LINE LENGTH)
The ADON function can't be used for reports of different lengths.

<ERR 697>(ORIGINAL REPORT DOES NOT EXIST)
Base report (the first RID specified) does not exist.

<ERR 700>(INVALID NUMBER OF LINES)
Correct and retry.

<ERR 705>(MAX NUMBER OF FIELDS EXCEEDED)
The F function cannot process more than 75 fields. Correct and retry.

<ERR 710>(INVALID OPTION)
Invalid character is in the option field of the F function. Correct and retry.

<ERR 711>(INVALID LINE TYPE)
Incorrect line type was specified. Only tab and * line type are allowed. Correct and retry.

<ERR 712>(INVALID PARAMETER FOR MASK)
Error occurs in deleting asterisks from the F function mask. Correct and retry.

<ERR 713>(INVALID EDIT–CODE IN RID 0)
Correct and retry.

<ERR 714>(NON–NUMERIC CHARACTER WAS SPECIFIED IN A NUMERIC FIELD)
Correct and retry.

<ERR 715>(NO PARAMETER)
Parameter must be entered in the mask of the FIND function. Correct and retry.

<ERR 716>(INVALID R–OPTION)
R option format must be

```
Rn-m
```

where:

`n-m`
Are RID numbers.

Correct and retry.

<ERR 717>(CORRESPONDING DATA DOESN'T EXIST)
No data matches the specified parameters.

<ERR 718>(NOT SORTED)
Report must be sorted before performing the BF function. Sort report data, then retry the BF function.

<ERR 719>(SPECIFIED REPORT MUST HAVE AN INDEX)
Reports processed by the BF function must have an index. To obtain an index, duplicate the original report; then delete the original report, replace it with the duplicated report, and retry.

<ERR 726>(INVALID OPTIONS)
Correct and retry.

<ERR 727>(INVALID PARAMETER FOR MASK)
Mask asterisks were incorrectly deleted. Correct and retry.

<ERR 728>(INVALID EDIT–CODE IN RID 0)
Correct and retry.

<ERR 729>(INVALID PARAMETER IN NUMERIC FIELD)
Specified numeric parameter is not in correct numeric format. Correct and retry.

<ERR 730>(INVALID PARAMETER FOR RANGE SEARCH)
Range search parameter *r* is incorrectly specified. Correct and retry.

<ERR 731>(MISSING SEARCH PARAMETER)
A parameter must be entered into the S function mask. Correct and retry.

<ERR 750>(DUPLICATE OPTION)
Same option is specified twice. Specify the option once and retry the function.

<ERR 751>(INVALID NUMERIC RANGE)

The S option must be in the following format:

```
Sx(-y)(,y)
```

where:

x,y
: Specifies the line range to scan.

Correct your line number range and retry.

<ERR 752>(INVALID T-OPTION)

T option must be in the following format:

```
Tx
```

where:

x
: Is the transparent character.

Correct and retry.

<ERR 753>(LINE NUMBER IS BEYOND LAST LINE)

S option specifies a line number beyond the last line. Correct and retry.

<ERR 754>(OPERATION STRING EXCEEDS 32 CHARACTERS)

Target or replacing string of the CHG function exceeds 32 characters. Correct and retry.

<ERR 755>(THERE IS NO OPERATION STRING)

Target and replacing string must both be specified in the CHG function. Correct and retry.

<ERR 760>(INVALID MASK IN ISSUING REPORT)

Asterisks incorrectly deleted in the sending report mask (the first mask). Correct and retry.

<ERR 761>(INVALID EDIT-CODE IN ISSUING REPORT)

Correct and retry.

<ERR 762>(INVALID MASK IN RECEIVING REPORT)
Asterisks were incorrectly deleted in the mask of the receiving report (the second mask). Correct and retry.

<ERR 763>(INVALID EDIT–CODE IN RECEIVING REPORT)
Correct and retry.

<ERR 764>(NUMBER OF MATCHING FIELDS OF BOTH REPORTS SHOULD BE THE SAME)
Correct and retry.

<ERR 765>(FIELD TYPE OF MATCHING FIELDS OF BOTH REPORTS SHOULD BE SAME)
Edit-codes of matching fields in the sending and receiving reports must be the same. Correct and retry.

<ERR 766>(LENGTH OF MATCHING FIELDS OF BOTH REPORTS SHOULD BE EQUAL)
Correct and retry.

<ERR 767>(SEQUENCE OF MATCHING FIELDS OF BOTH REPORTS ARE NOT SAME)
Matching fields of the sending and receiving reports have different sorted types.

<ERR 768>(NUMBER OF MOVING FIELDS OF BOTH REPORTS SHOULD BE EQUAL)
Correct and retry.

<ERR 769>(INVALID OPTIONS)
Valid options for the MA function are: C, D, F, I, M, N, P, and S. Correct and retry.

<ERR 770>(INVALID CHARACTER IN ISSUING FIELD)
Correct and retry.

<ERR 771>(DUPLICATE PARAMETER IN ISSUING FIELDS)
Duplicate matching levels (1–5) are incorrectly specified in the issuing report mask. Correct and retry.

<ERR 772>(INVALID PARAMETER IN ISSUING FIELD)
Matching levels incorrectly specified in the issuing report mask. Valid levels are 1 through 5. Correct and retry.

<ERR 773>(MISSING MATCHING FIELD IN ISSUING REPORT)
Matching field must be specified in the issuing report mask. Correct and retry.

<ERR 774>(MATCHING LEVELS NOT CONSECUTIVE IN ISSUING REPORT)
Matching levels must be specified as follows: 1, 2, 3, 4, 5. Correct and retry.

<ERR 775>(INVALID LENGTH OF ISSUING FIELD)
Length of matching field in the issuing report exceeds 36 characters. Correct and retry.

<ERR 776>(DUPLICATE PARAMETER IN ISSUING FIELDS)
A moving field parameter is repeated. Valid parameters are specified as follows:

```
A,B,C,...,M
```

Correct and retry.

<ERR 777>(MISSING MOVING FIELD IN ISSUING REPORT)
Correct and retry.

<ERR 778>(MOVING FIELDS NOT CONSECUTIVE IN ISSUING REPORT)
Moving fields must be specified in the following order: A,B,C,...,M. Correct and retry.

<ERR 779>(DATA OF ISSUING REPORT NOT SORTED)
The P option was specified in the MA function, but data had not been presorted. Presort data and retry MA function with P option.

<ERR 780>(INVALID PARAMETER IN RECEIVING REPORT)
Correct and retry.

<ERR 781>(DUPLICATE MATCHING KEY IN RECEIVING REPORT)
A matching level parameter cannot be repeated twice. Only five matching levels are permitted. Valid levels are 1, 2, 3, 4, 5. Correct and retry.

<ERR 782>(INVALID MATCHING LEVEL IN RECEIVING REPORT)
Valid matching levels are: 1, 2, 3, 4, 5. Specify them consecutively. Correct and retry.

<ERR 783>(MISSING MATCHING FIELD IN RECEIVING REPORT)
Correct and retry.

<ERR 784>(INVALID MATCHING KEY IN RECEIVING REPORT)
Matching levels must be specified consecutively. Valid levels are 1, 2, 3, 4, 5. Correct and retry.

<ERR 785>(INVALID LENGTH OF MATCHING KEY IN RECEIVING REPORT)
Total length of the matching field exceeds 36 characters. Correct and retry.

<ERR 786>(DUPLICATE MOVING FIELD IN RECEIVING REPORT)
A moving field parameter cannot be repeated twice. Valid parameters are:

```
A,B,C,...,M.
```

Specify parameters consecutively. Correct and retry.

<ERR 787>(MISSING MOVING FIELD IN RECEIVING REPORT)
Correct and retry.

<ERR 788>(MOVING FIELD NOT CONSECUTIVE IN RECEIVING REPORT)
Moving field parameters must be specified in consecutive order. Valid parameters are: A, B, C, ..., M. No duplications are allowed. Correct and retry.

<ERR 789>(DATA IN RECEIVING REPORT NOT SORTED)
The P option is specified in the MA function but data in the report is not presorted. Presort data and retry the MA function with the P option.

<ERR 790>(SYSTEM ERROR IN SORTING ISSUING REPORT (MPDP))
Error occurred in MPDP while sorting the report. See MAPPER 80 coordinator or take a job dump.

<ERR 791>(SYSTEM ERROR IN SORTING ISSUING REPORT (SORT))
Error occurred in FPRSOW while sorting the report. See MAPPER 80 coordinator or take a job dump.

<ERR 792>(SYSTEM ERROR IN SORTING RECEIVING REPORT (MPDP))
Error occurred in MPDP while sorting the report. See MAPPER 80 coordinator or take a job dump.

<ERR 793>(SYSTEM ERROR IN SORTING RECEIVING REPORT (SORT))
Error occurred in FPRSOW while sorting the report. SEE MAPPER 80 coordinator or take a job dump.

<ERR 794>(SYSTEM ERROR IN SORTING MATCHED RESULT (MPDP))
Error occurred in MPDP while sorting the report. See MAPPER 80 coordinator or take a job dump.

<ERR 795>(SYSTEM ERROR IN SORTING MATCHED RESULT (SORT))
Error occurred in FPRSOW while sorting the result. See MAPPER 80 coordinator or take a job dump.

<ERR 800>(SYSTEM ERROR: EDIT–CODE OF RID 0 IS BAD)
See MAPPER 80 coordinator or take a job dump.

<ERR 801>(INVALID CHARACTER IN OPTION FIELD)
Valid options for TOT are: A,C,E,H,I,O,Rn,=X,*.
Correct and retry.

<ERR 802>(INVALID R,U,D–OPTION)
Correct option formats are as follows:

```
Rx-y(Rx,y)
```
where x,y are selected RIDs.

```
U or U(x)
```
where x is a search RID.

D

Enter all options on the line directly above the mask. Correct and retry.

<ERR 803>(CANNOT ENTER CHARACTER INTO FIRST COLUMN OF MASK)
Correct and retry.

<ERR 804>(PARAMETERS SHOULD BE SPECIFIED IN FIRST COLUMN OF FIELD)
Correct and retry.

<ERR 805>(CANNOT INPUT CHARACTER AFTER "M,S,C,A")
Parameters M, S, C, and A cannot have trailing characters. Correct and retry.

<ERR 806>(INVALID NUMERIC DATA)
Trailing characters not in correct numeric format. Correct and retry.

<ERR 807>(DID NOT ENTER PARAMETER ON FIRST LINE OF MASK PARAMETER AREA)
Correct and retry.

<ERR 808>(ONLY THE FOLLOWING PARAMETERS CAN BE SPECIFIED MORE THAN ONCE: + – = A)
Correct and retry.

<ERR 809>(CANNOT SPECIFY "+" AND "/" SIMULTANEOUSLY)
Correct and retry.

<ERR 810>(ALL "M" PARAMETERS MUST HAVE "=" PARAMETERS)
Correct and retry.

<ERR 811>(CANNOT DO MOVE BETWEEN KANJI AND NON–KANJI FIELDS)
Correct and retry.

<ERR 812>(CANNOT PUT KANJI–DATA IN ALPHANUMERIC FIELD)
Correct and retry.

<ERR 813>(CANNOT PUT ALPHA–NUMERIC DATA IN A KANJI FIELD)
Correct and retry.

<ERR 814>(INVALID COMBINATION OF OPTION AND PARAMETER)
Specified option/parameter combination is incorrect for horizontal arithmetic. Correct and retry.

<ERR 815>(CANNOT HAVE MORE THAN 16 + PARAMETERS)
Remove + sign and retry.

<ERR 816>(INVALID OPTIONS COMBINATION (R,U,D))
Specified option/parameter combination is incorrect for vertical summation. Correct and retry.

<ERR 817>(CANNOT DESIGNATE KANJI–ITEM AS ARITHMETIC DATA)
Specified option/parameter combination is incorrect for subtotaling. Correct and retry.

<ERR 818>(CANNOT SPECIFY VARIABLES ON @TOT EXCEPT AVERAGING & VERTICAL SUMM.)
Correct and retry.

<ERR 819>(INVALID PARAMETERS COMBINATION)
Correct and retry.

<ERR 820>(INVALID CHARACTER IN PARAMETER)
Correct and retry.

<ERR 821>(INVALID PARAMETER INPUT – HORIZONTAL ARITHMETIC–)
Must be trailer data after * or / when * or / and = combination is specified, e.g., *100 =

<ERR 822>(LENGTH OF KEY–FIELD SHOULD BE WITHIN 24 CHARACTERS)
Correct and retry.

<ERR 823>(TOTAL NO. OF PARAMETERS SHOULD NOT BE GREATER THAN 18)
Reduce number of parameters and retry.

<ERR 824>(LENGTHS OF MOVING FIELDS IN BOTH MASKS DO NOT MATCH)
Lengths of moving fields in the issuing and receiving reports must be equal. Correct and retry.

<ERR 825>(FILLING FIELD CANNOT SPECIFY OPTIONS)
Filling or moving fields cannot specify options. Correct and retry.

<ERR 826>(INVALID * DELETION IN MASK)
Correct and retry.

<ERR 827>(ONLY VERTICAL SUMMATION PARAMETER IS ALLOWED ON 2ND LINE)
Parameters specified on the second parameter line are for vertical summation only. Correct and retry.

<ERR 828>(INVALID COMBINATION OF = AND ANOTHER PARAMETER)
Specified parameter cannot be combined with an equal sign (=). Correct and retry.

<ERR 829>(INVALID COMBINATION OF PARAMETERS ON LINES 1 AND 2)
Combination of horizontal and vertical summation is specified by two lines of parameters. Correct and retry.

<ERR 830>(SYSTEM ERROR OCCURRED (TOT))
Error occurred in the FPRTOT module while analyzing parameters. See MAPPER 80 coordinator or take a job dump.

<ERR 855>(CANNOT DELETE MODE WHEN IN USE)
A mode can't be deleted (using MDEL) when the mode is in use. Make sure there are no types marked open.

<ERR 856>(INVALID MODE NUMBER)
Mode specified in the MDEL function is incorrect. Correct and retry.

<ERR 857>(ODD MODE NUMBER IS INVALID)
Specified mode must be even for the following functions:

```
MGEN
MMOD
MDSP
MDEL
```

<ERR 858>(MUST INPUT MODE NAME)
Mode name must be specified in the MGEN function. Correct and retry.

<ERR 859>(MODE NUMBER DOESN'T EXIST)
Correct and retry.

<ERR 860>(SPECIFIED TYPE ALREADY EXISTS)
Type specified in the TGEN function already exists. Correct and retry.

<ERR 861>(INVALID TYPE)
Type specified in the TDEL function already in OPEN status.

<ERR 862>(CANNOT DELETE TYPE WHEN IN USE)
Specified type can't be deleted because RID in the type is in use.

<ERR 864>(INVALID START COLUMN NUMBER)
Incorrect starting column specified for the print. Correct and retry.

<ERR 865>(INVALID DEVICE NAME)
Specified output device name is invalid. Correct and retry.

<ERR 866>(CANNOT USE THIS REPORT)
Specified report cannot be printed.

<ERR 867>(CANNOT REGISTER AUXILIARY DEVICE. PLEASE RE-TRY.)
Requested AUX function is not processed. Retry.

<ERR 868>(NO FREE RIDS LEFT IN PRINT QUEUE)
No free RIDs are left in AUX queue. See MAPPER 80 coordinator.

<ERR 869>(REPORT ERROR (FMT))
Errors occurred in RID 0 of the result being printed. Correct and retry.

<ERR 870>(REPORT ERROR (GL))
System error occurred in a MPDP request (GL). See MAPPER 80 coordinator or take a job dump.

<ERR 871>(REPORT ERROR (DR))
System error occurred in a MPDP request (DR). See MAPPER 80 coordinator or take a job dump.

<ERR 872>(REPORT ERROR (GF))
System error occurred in a MPDP request (GF). See MAPPER 80 coordinator or take a job dump.

<ERR 873>(AUXILIARY DEVICE ISN'T CONNECTED)
Specified output COP device is not connected to job.

<ERR 874>(NO FREE AUXILIARY DEVICE. PLEASE RE-TRY)
Specified auxiliary printer is in use. Retry after short wait.

<ERR 875>(THERE IS NO REPORT FOR MASK)
Specified RID doesn't exist when displaying mask.

<ERR 876>(THERE IS NO ISSUING REPORT FOR MASK)
Correct and retry.

<ERR 877>(INVALID MODE PASSWORD FOR ISSUING REPORT)
Correct and retry.

<ERR 880>(THIS MODE IS ALREADY CATALOGUED)
Mode number specified in the MGEN function already exists. You cannot generate same mode number twice.

<ERR 881>(CANNOT DELETE CURRENT MODE)
Mode number specified in MDEL function cannot be deleted.

<ERR 885>(INVALID OPTION)
Error occurred in the option specification. Correct and retry.

<ERR 886>(INVALID FIELD PARAMETER)
Incorrect parameter is in the input. Correct and retry.

<ERR 887>(THERE IS NO PARAMETER IN FIELD (x))
Parameter must be specified in field (x). Correct and retry.

<ERR 888>(THERE IS NO PARAMETER IN FIELD (1))
Parameter must be specified in field (1). Correct and retry.

<ERR 889>(CANNOT SPECIFY PARAMETER IN FIELD (2))
Correct and retry.

<ERR 890>(CANNOT SPECIFY PARAMETER IN FIELD (3))
Correct and retry.

<ERR 891>(INVALID PARAMETER IN FIELD (2,3))
Correct and retry.

<ERR 892>(CANNOT USE THIS REPORT)
There are no data lines in this report.

<ERR 893>(MUST BE NUMERIC)
Report does not contain numeric data. The G function cannot be performed on a report unless numeric data is present.

<ERR 894>(PRIMITIVE DATA IS INCORRECT)
Correct data and retry functions.

<ERR 895>(INVALID U-OPTION)
The line specified by the U-option is not a comment line. Correct and retry.

<ERR 896>(FIELD LENGTH EXCEEDS 12 CHARACTERS)
Correct and retry.

<ERR 897<(INVALID OPTION)
Invalid option was entered. Correct and retry.

<ERR 900>(RUN IS NOT REGISTERED)
Specified run is not registered in the run-list.

<ERR 901>(RUN–REPORT DOES NOT EXIST)
No run report exists with the specified name.

<ERR 902>(INVALID LABEL)
Correct and retry.

<ERR 903>(DUPLICATE LABEL)
Duplicate labels are in the run. Change one of the labels and retry.

<ERR 904>(THERE ARE NO DATA LINES IN THE RUN REPORT)
Run report contains only header lines.

<ERR 905>(CANNOT USE THIS MODE FOR THIS RUN)
Specified mode cannot be used for this run. Correct and retry.

<ERR 906>(REPORT IN ODD–NUMBERED MODE CANNOT BE UPDATED)
Only reports in even-numbered modes can be updated.

<ERR 907>(MODE DOES NOT EXIST)
Try another mode. Rerun.

<ERR 908>(TYPE DOES NOT EXIST)
Try another type. Rerun.

<ERR 909>(RID DOES NOT EXIST)
Try another RID. Rerun.

<ERR 910>(RESULT DOES NOT EXIST)
No result has been created.

<ERR 911>(LINE NUMBER DOES NOT EXIST IN REPORT)
Specified data line is not a valid line in the report. Try another line and rerun.

<ERR 912>(REQUIRED PARAMETER MISSING)
The following parameters are required for the @SUB run function:

s

Specifies the key field.

+

Specifies the fields to be subtotaled.

Correct and retry.

<ERR 913>(INVALID DELIMITER)
Delimiter character is invalid. Correct and retry.

<ERR 914>(CANNOT USE RESERVED WORD FOR PARAMETER)
First enter the reserved word in a variable (e.g., @CHG V1A4 reserved-word). Then use the variable as a parameter. Correct and retry.

<ERR 915>(PARAMETER TOO LONG)
Correct and retry.

<ERR 916>(MUST BE NUMERIC)
Nonnumeric characters were entered in a designated numeric field. Correct and retry.

<ERR 917>(INVALID TYPE)

An incorrect type was specified. Valid types are:

- A (free-form type)
- B thru I

Correct and retry.

<ERR 918>(MUST BE ALPHABETIC)

Nonalphabetic characters were entered in a designated alphabetic field. Correct and retry.

<ERR 919>(NUMERIC CHARACTER MUST BE RIGHT JUSTIFIED)

A numeric character cannot be entered into a field with trailing blanks.

```
Invalid:   156ΔΔ→field x
Valid:     ΔΔ156→field x
```

Correct and retry.

<ERR 920>(NO BLANKS ALLOWED IN NUMERIC FIELD)

Correct and retry.

<ERR 921>(MUST BE NUMERIC)

You cannot enter only a minus (–) into a numeric field. The minus must be followed by a number. Correct and retry.

<ERR 922>(MUST BE NUMERIC)

A minus (–) character is placed in the middle of the numeric string. It must be before the number.

```
Invalid: 2-3→field x
Valid:   -23→field x
```

Correct and retry.

<ERR 923>(CANNOT USE "–" IN NUMERIC PARAMETER)

The minus (–) character cannot be entered into the specified numeric parameter. Correct and retry.

<ERR 924>(MUST BE NUMERIC)

Nonnumeric characters were entered into a designated numeric field. Correct and retry.

<ERR 925>(INVALID LABEL (NOT 1–999))
All labels must be in the range 1 through 999. Correct and retry.

<ERR 926>(UNDEFINED LABEL)
You are referencing a label that does not exist. The format of the label definition is shown in the following example:

```
@1ØØ:CHG V1 Ø
```

Correct and retry.

<ERR 927>(REPORT IS LOCKED)
The report you are trying to access is locked by another user. Retry after a short wait.

<ERR 928>(ERROR OCCURRED. CANNOT DISPLAY RUN REPORT BECAUSE OF I/O ERROR)
An error occurred in the run. This caused an error in MPDP when trying to display the run report. See MAPPER 80 coordinator.

<ERR 929>(CANNOT USE DEFINED VARIABLE FOR PARAMETER)
A newly-defined variable cannot be entered into the first field of the run statement. Define the variable before using it. Retry run.

<ERR 930>(REPORT REACHED MAX LINES)
The report has 99999 lines. No more lines can be added to this report. Delete lines or use another report.

<ERR 931>(CANNOT ADD LINES BECAUSE MAX NUMBER OF LINES EXCEEDED)
The report will reach the maximum number of lines after execution of the @LNX or @LN+ run function. Delete lines or use another report.

<ERR 932>(OBJECT REPORT IS LOCKED)
The report specified in the @LN+ run function is in use. Retry after a moment.

<ERR 933>(INVALID NO. OF LINES)
The number of duplicated lines specified in the @LNX run function cannot exceed 23. Correct and retry.

<ERR 934>(INVALID NO. OF TIMES)

The number of times to duplicate the lines cannot be greater than 99. Correct and retry.

<ERR 935>(NO. OF ADDITION LINES EXCEEDS 99 LINES)

The total number of lines added by the @LNX run function cannot be greater than 99.

$$\text{total lines added} = \boxed{\begin{array}{l}\text{number of}\\ \text{duplicated}\\ \text{lines}\end{array}} \times \boxed{\begin{array}{l}\text{number of}\\ \text{times to}\\ \text{duplicate}\end{array}}$$

Correct and retry.

<ERR 936>(INVALID NO. OF DELETE LINES)

You cannot delete more than 999 lines at a time. Correct and retry.

<ERR 937>(NEXT LINE MUST BE OUTPUT LINE)

An output line must follow the @RDC and @SUB run functions. Correct and retry.

<ERR 938>(NO OTHER RUN COMMAND CAN FOLLOW THIS RUN COMMAND ON THE SAME LINE)

The following run functions cannot have another run function on the same line with them:

```
@DSP,@ESR,@RSR,@RDC,@RUN,@SUB
```

Enter the second run function on the line below the first. Rerun.

<ERR 939>(END OF RUN REPORT)

You have reached the last line of the run report. No more lines can be added to this report.

<ERR 940>(EXCEEDED 256 CHARACTERS IN TOTAL RUN COMMAND)

The length of a run function cannot be greater than 256. Correct and retry.

<ERR 941>(NO. OF CONTINUATION LINES EXCEEDS 10)

The number of continuation lines of any run function cannot be greater than 10. Correct and retry.

<ERR 942>(NO DATA IN CONTINUATION LINE)
A continuation line must contain data. Correct and retry.

<ERR 943>(MUST HAVE RUN COMMAND FOLLOWING ;)
Correct and retry.

<ERR 944>(INVALID RUN COMMAND NAME)
Invalid run function

<ERR 945>(CANNOT TERMINATE FIELD BY (COMMA,/,–))
Correct and retry.

<ERR 946>(TOO MANY SUB-FIELDS)
Correct and retry.

<ERR 947>(INVALID EXPRESSION FOR LITERAL (NO. OF QUOTES MUST BE EVEN))
When describing literal strings, quotes must appear in pairs. Correct and retry.

<ERR 948>(LACK OF FIELD)
Fields are missing in the run function. Correct and retry.

<ERR 949>(TOO MANY FIELDS)
Too many fields are specified in the run function. Correct and retry.

<ERR 950>(LACK OF SUB–FIELD)
A required subfield is missing from the run function. Correct and retry.

<ERR 951>(TOO MANY SUB–FIELDS)
Too many subfields specified in the run function. Correct and retry.

<ERR 952>(NO INITIAL VALUE FOR VARIABLE)
There is no initial value given for this variable. Use either the colon (:) statement or the @CHG run function to initialize the variable before using it. Correct and retry.

<ERR 953>(INVALID LENGTH OF INITIAL VALUE)
The length of the initial value for the variable must equal the designated variable length. Correct and retry.

<ERR 954>(INVALID DEFINITION OF VARIABLE)
The correct variable definition formats are:

1. Colon definition

```
e.g., :V1A3[ABC ;V2H3[DEF
```

2. @CHG definition

```
e.g., @CHG V3F5.2 12.34
e.g., @RDL,2,B,6,6  F1,F2  V1ØA2,V11A6
```

3. Run function definition

```
e.g., @RDL,2,B,6,6  F1,F2  V1ØA2,V11A6
```

Correct and retry.

<ERR 955>(INVALID USE OF VARIABLE)
A variable was used incorrectly in either the colon (:) line or the output line. Correct and retry.

<ERR 956>(NO FREE SPACE IN VARIABLE AREA)
No more space exists in the variable area. No more variables can be defined in the run.

Redesign the run so that extraneous variables are deleted. Then, define new variables.

<ERR 957>(INVALID VARIABLE NUMBER)
All variable numbers must be in the range V1 thru V999. Correct and retry.

<ERR 958>(INVALID VARIABLE TYPE)

The following types of variables are available:

A	Alphanumeric ≤16 characters
H	Alphanumeric and special characters ≤16 characters
I	Integers ≤16 characters
F	Real numbers ≤16 characters, 14 of which can be fractional
S	Alphanumeric and special characters ≤132

Correct and retry.

<ERR 959>(INVALID LENGTH OF VARIABLE)

All variable types can be up to 16 characters in length, except for type S variables, which can be up to 132 characters. Correct and retry.

<ERR 960>(CAN'T REFER TO UNDEFINED VARIABLE)

Define the variable before using it or referring to it. Variables can be defined with the colon (:) statement, the @CHG run function or in the @RUN statement itself. Correct and retry.

<ERR 961>(INVALID POSITION OF START CHARACTER)

An undefined variable cannot be used to specify the start column number of a partial variable. Correct and retry.

<ERR 962>(INVALID VARIABLE TYPE (NOT "A","I"))

The variable that specifies the start column of a partial variable must be type A or I. Correct and retry.

<ERR 963>(INVALID SPECIFICATION OF START CHARACTER POSITION)

The start column number of the partial variable is incorrectly specified. It must either be a number or a previously initialized variable of type A or I. Correct and retry.

<ERR 964>(UNDEFINED VARIABLE CANNOT SPECIFY START CHARACTER POSITION)

Define the start column variable before using it (define it with colon (:) statement or @CHG run function). Retry.

<ERR 965>(CANNOT SPECIFY LENGTH BECAUSE OF INVALID VARIABLE NO.)
An undefined variable cannot be used to specify the length of a partial variable. Define it before using it (with the colon (:) statement or the @CHG run function). Correct and retry.

<ERR 966>(CANNOT SPECIFY LENGTH BECAUSE OF BAD VARIABLE TYPE (NOT "A" OR "I"))
The variable specifying the length of the partial variable must be of type A or I. Correct and retry.

<ERR 967>(CANNOT SPECIFY LENGTH BECAUSE OF INVALID VARIABLE)
The length of the partial variable is incorrectly specified. It must be either a number or a previously initialized variable of types A or I. Correct and retry.

<ERR 968>(CANNOT SPECIFY LENGTH BECAUSE OF UNDEFINED VARIABLE)
Define the length variable before using it with the colon (:) statement or the @CHG run function. Retry.

<ERR 969>(DEAD LOCK OCCURRED)
System error occurred when reading run report from WRKR. Retry after a short wait. If same error occurs, see MAPPER 80 coordinator.

<ERR 970>(MPDP ERROR OCCURRED WHEN READING REPORT)
System error occurred when reading run report from WRKR. See MAPPER 80 coordinator.

<ERR 971>(CANNOT USE "@" AND ":" AT THE HEAD OF A CONTINUATION LINE).
Correct and retry.

<ERR 972>(CANNOT USE RESERVED WORD, VARIABLE AND LITERAL IN THIS PARAMETER)
Correct and retry.

<ERR 973>(CANNOT USE VARIABLE AND RESERVED WORD IN OPTION FIELD)
Options must be specified as character strings or literals. Correct and retry.

<ERR 974>(UNNECESSARY PARAMETER MUST BE OMITTED)
Correct and retry.

<ERR 975>(REQUIRED PARAMETER WAS OMITTED)
Correct and retry.

<ERR 976>(THIS PARAMETER MUST BE A VARIABLE)
Correct and retry.

<ERR 977>(CANNOT USE PARTIAL VARIABLE)
Correct and retry.

<ERR 978>(INVALID VARIABLE TYPE)
Permissible variable types are A, I, or F. Correct and retry.

<ERR 979>(TOO MANY PARAMETERS)
Valid parameters for the SUB command are:

+ΔSΔM

A maximum of 16 for each can be specified. Correct and retry.

<ERR 980>(INVALID CHARACTER IN ARITHMETIC PARAMETER)
Correct and retry.

<ERR 981>(PARAMETER "S" SHOULD BE THE ONLY ONE)
Correct and retry.

<ERR 982>(PARAMETER "S" IS REQUIRED)
Correct and retry.

<ERR 983>(MAXIMUM NUMBER OF PARAMETER "+","M" IS 16)
Correct and retry.

<ERR 984>(KEY FIELD MUST BE WITHIN 24 CHARACTERS)
The length of the key field must be less than or equal to 24 characters. Correct and retry.

<ERR 985>(INVALID LINE TYPE)
Correct and retry.

<ERR 986>(CANNOT USE KANJI DATA FOR SUBTOTALING)
Correct and retry.

<ERR 987>(NO "+" PARAMETER)
Specify + parameter and retry.

<ERR 988>(TOO MANY VARIABLES)

The specification of variables for the @SUB run function is as follows.

- First variable (with E option):

  number of lines of each subgroup

- Second and subsequent variables:

  key field (s), subtotal value (+), and data of last line of subgroup (M)

A variable must be specified for each parameter plus one extra if the E option is specified. Correct and retry.

<ERR 989>(LACK OF VARIABLES)
See explanation of <ERR 988>.

<ERR 990>(DIFFERENT TYPE BETWEEN VARIABLE AND DATA)
Kanji data is used incorrectly in the @SUB run function. Correct and retry.

<ERR 991>(INVALID LENGTH OF VARIABLE)
The variable corresponding to the S or M parameter must have the same length as the specified field. Correct and retry.

<ERR 992>(DIFFERENT NO. OF VARIABLE BETWEEN "SUB" AND OUTPUT LINE)
The number of variables in the output line following the @SUB run function must equal the number of variables specified in the @SUB run function.

<ERR 993>(SYSTEM ERROR: LOGIC PROBLEM IN RUN CONTROL STATEMENT)
An error occurred while analyzing the run control statement. See MAPPER 80 coordinator.

<ERR 994>(INVALID CHARACTER FIELD)
Correct and retry.

<ERR 995>(PERIOD LINES CANNOT BE LONGER THAN 80 CHARACTERS)
Correct and retry.

<ERR 996>(FIELD NUMBER TOO LARGE)

When specifying the object field in the format Fnn, nn should be the existing field number. This number can be obtained with the FLD function. Correct and retry.

<ERR 997>(INVALID KANJI FIELD)

Correct and retry.

<ERR 998>(INVALID KANJI FIELD. FIRST EDIT-CODE MUST BE AN EVEN NUMBER)

Correct and retry.

<ERR 999>(BAD KANJI FIELD. DIFFERENT EDIT–CODE CAN'T BE MIXED IN SAME FIELD)

Correct and retry.

Index

CUT

SPERRY

USER COMMENT SHEET

We will use your comments to improve subsequent editions.

NOTE: Please do not use this form as an order blank.

(Document Title)

(Document No.)	*(Revision No.)*	*(Update No.)*

Comments:

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

CUT

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY CORPORATION

ATTN.: SOFTWARE SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

FOLD

CUT

SPERRY

USER COMMENT SHEET

We will use your comments to improve subsequent editions.

NOTE: Please do not use this form as an order blank.

(Document Title)

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

CUT

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY CORPORATION

ATTN.: SOFTWARE SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

FOLD