



**PUBLICATIONS
UPDATE**

Operating System/3 (OS/3)

**Screen Format Services
Concepts and Facilities**

UP-9977-B

This Library Memo announces the release and availability of Update B to "SPERRY® Operating System/3 (OS/3) Screen Format Services Concepts and Facilities", UP-9977.

The OS/3 screen format services let programmers create and maintain formatted screen displays for use with their application programs, as well as with COBOL, RPG II, FORTRAN IV, and assembler programs.

This update documents for release 10.0 the enhancements providing lowercase constant translation on the spool option and lowercase to uppercase translation at the field level. All other changes are expanded descriptions, corrections, or clarifications applicable to features present in the software before release 10.0.

Copies of Update B are now available. You can order the update only or the complete manual with the update through your local Sperry representative. To receive only the update, order UP-9977-B. To receive the complete manual, order UP-9977.

LIBRARY MEMO ONLY

Mailing Lists
BZ, CZ, and MZ

LIBRARY MEMO AND ATTACHMENTS

Mailing Lists AF01, B00, B01,
MBW, 28U, and 29U
(Package B to UP-9977,
22 pages plus Memo)

THIS SHEET IS

Library Memo for
UP-9977-B

RELEASE DATE:

May 1986





**PUBLICATIONS
UPDATE**

Operating System/3 (OS/3)

Screen Format Services

Concepts and Facilities

UP-9977-A

This Library Memo announces the release and availability of Updating Package A to "SPERRY® Operating System/3 (OS/3) Screen Format Services Concepts and Facilities", UP-9977.

The OS/3 screen format services permit programmers to create and maintain formatted screen displays for use with their application programs, as well as with COBOL, RPG II, FORTRAN IV, and assembler programs.

This update documents for release 9.0 the new Batch Format Generator feature. All other changes are expanded descriptions, corrections, or clarifications of the 8.2 release.

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-9977-A. To receive the complete manual, order UP-9977.

LIBRARY MEMO ONLY

Mailing Lists
BZ, CZ, and MZ

LIBRARY MEMO AND ATTACHMENTS

Mailing Lists B00, B01, 28U, and 29U
(Package A to UP-9977
22 pages plus Memo)

THIS SHEET IS

Library Memo for
UP-9977-A

RELEASE DATE:

January, 1985



Screen Format Services



Environment: System 80

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry representative.

Sperry reserves the right to modify or revise the content of this document. No contractual obligation by Sperry regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry.

FASTRAND, ✦SPERRY, SPERRY, SPERRY✦UNIVAC, SPERRY UNIVAC, UNISCOPE, UNISERVO, UNIVAC, and ✦ are registered trademarks of the Sperry Corporation. ESCORT, MAPPER, PAGEWRITER, PIXIE, SPERRYLINK, and UNIS are additional trademarks of the Sperry Corporation.

PAGE STATUS SUMMARY

ISSUE: Update B – UP-9977
RELEASE LEVEL: 10.0 Forward

| Part/Section | Page Number | Update Level |
|------------------|-------------|--------------|
| Cover/Disclaimer | | Orig. |
| PSS | 1 | B |
| Preface | 1 thru 3 | Orig. |
| Contents | 1 thru 3 | Orig. |
| | 4, 5 | A |
| | 6 | Orig. |
| 1 | 1 thru 13 | Orig. |
| 2 | 1 thru 3 | Orig. |
| | 4 | B |
| | 5, 6 | Orig. |
| | 7 | B |
| 3 | 1, 2 | Orig. |
| | 3, 4 | B |
| | 5 | A |
| | 6 thru 34 | Orig. |
| | 35 thru 38 | B |
| | 39 | A |
| 4 | 40 thru 53 | Orig. |
| | 1 thru 21 | Orig. |
| | 22 | B |
| | 23 | Orig. |
| | 24 | B |
| 5 | 25 thru 32 | Orig. |
| | 1 thru 5 | Orig. |
| | 6 | A |
| 6 | 7 | A |
| | 1 thru 10 | Orig. |
| | 11, 12 | B |
| | 13, 14 | Orig. |
| | 15 | A |
| 7 | 16 | Orig. |
| | 1 thru 34 | Orig. |
| | 35, 36 | A |
| Appendix A | 37, 38 | Orig. |
| | 1 thru 7 | Orig. |
| | Appendix B | 1 thru 3 |
| 4 | | B |
| 5 thru 7 | | Orig. |
| 8 | | B |
| Appendix C | 1 | Orig. |
| Appendix D | 1 | Orig. |
| Appendix E | 1 | Orig. |

| Part/Section | Page Number | Update Level |
|--------------------|-------------|--------------|
| Appendix F | 1 thru 13 | Orig. |
| Appendix G | 1, 2 | A |
| Index | 1 | A |
| | 2 thru 17 | Orig. |
| User Comment Sheet | | |

| 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 programmer in the use of the SPERRY Operating System/3 (OS/3). It specifically describes screen format services (SFS), the interactive component of OS/3 that provides an easy and effective method for inputting and outputting data.

The intended audience is the programmer with a basic knowledge of data processing, but with little programming experience.

This manual is divided as follows:

■ Section 1. Introduction

- Introduces you to screen formats as an input and output facility for programs.
- Describes the capabilities that SFS provides, via the screen format generator (SFG) and the screen format coordinator (SFC), for creating and using screen formats.

■ Section 2. Type Attributes and Editing Rules

Discusses a variation of COBOL editing rules to be observed when creating your own formats.

■ Section 3. Using the Screen Format Generator (SFG)

- Explains how to initiate the SFG session.
- Describes the home screen.
- Discusses the use of the HELP function.
- Describes the characteristics screen and the results produced by making various specifications on it.
- Explains the various dialog and optional screens.

- Section 4. Creating Screen Formats

Provides detailed instructions for creating your own screen formats.

- Section 5. Modifying Formats

Explains various methods for changing existing formats.

- Section 6. Additional Functions

Describes the functions that enable you to display and print existing formats and format information. Explains the functions that delete formats and terminate the SFG session.

- Section 7. Associating Formats with a Program: Using the Screen Format Coordinator (SFC)

- Explains how screen formats become functional with a program through the SFC.
- Discusses job control requirements.
- Introduces programming requirements related to RPG II, COBOL, FORTRAN IV, assembler language, ESCORT, and IMS interfaces.

- Appendix A. Job Control Considerations

Provides more information about the USE SFS job control statement. Describes the job control language statements necessary to use more than one workstation (applicable to RPG II programmers only).

- Appendix B. The 12-Line Screen Display

Shows the 12-line screen display format for operators using a terminal with 12-line capability.

- Appendix C. Function Keys

Describes the various function keys you can use with screen format services.

The following publications are referenced in this manual and are either useful or necessary to the programmer working with screen format services.

- Consolidated data management macro language user guide/programmer reference, UP-9979
- Assembler user guide, UP-8913
- FORTRAN IV programmer reference, UP-8814

- Interactive services commands and facilities user guide/programmer reference, UP-9972
- RPG II user guide, UP-8067
- 1974 ANSI COBOL programmer reference, UP-8613
- Job control user guide, UP-9986
- General editor user guide/programmer reference, UP-9976



Contents

PAGE STATUS SUMMARY

PREFACE

CONTENTS

1. INTRODUCTION

| | | |
|----------|---|------|
| 1.1. | PROVIDING INFORMATION VIA FORMS | 1-1 |
| 1.2. | PROVIDING INFORMATION VIA SCREEN FORMAT SERVICES | 1-1 |
| 1.3. | SCREEN FORMAT GENERATOR | 1-6 |
| 1.3.1. | Functions of the SFG | 1-6 |
| 1.3.2. | Screen Format Components | 1-8 |
| 1.3.2.1. | Display Constants | 1-9 |
| 1.3.2.2. | Variable Data Fields | 1-9 |
| 1.3.2.3. | Protected and Unprotected Parts – Low-Intensity Display | 1-11 |
| 1.4. | SCREEN FORMAT COORDINATOR (SFC) | 1-12 |

2. TYPE ATTRIBUTES AND EDITING RULES

| | | |
|--------|--|-----|
| 2.1. | CONSIDERATIONS FOR VARIABLE DATA FIELDS | 2-1 |
| 2.2. | TYPE ATTRIBUTES | 2-1 |
| 2.3. | EDITING | 2-2 |
| 2.3.1. | Editing Alphabetic and Alphanumeric Fields | 2-2 |
| 2.3.2. | Editing Numeric Fields | 2-3 |
| 2.4. | RESULTS OF EDITING FORMAT FIELDS | 2-4 |

3. USING THE SCREEN FORMAT GENERATOR (SFG)

| | | |
|---------|---|------|
| 3.1. | ACTIVATING THE SFG | 3-1 |
| 3.2. | THE HOME SCREEN | 3-2 |
| 3.2.1. | Functions Provided by the SFG | 3-3 |
| 3.2.2. | New Format Name and Library | 3-4 |
| 3.2.3. | Old Format Name and Library | 3-5 |
| 3.3. | THE CHARACTERISTICS SCREEN | 3-6 |
| 3.3.1. | Error Retry Counts (Line 3) | 3-7 |
| 3.3.2. | Alphabet (Line 4) | 3-7 |
| 3.3.3. | Lowercase Translation (Line 6) | 3-7 |
| 3.3.4. | Original and Overlay Formats (Line 8) | 3-8 |
| 3.3.5. | Erasing and Unlocking Options (Lines 10, 11, and 12) | 3-8 |
| 3.3.6. | Transmit All-Disregard Cursor Position (Line 14) | 3-10 |
| 3.3.7. | Special Editing Characters (Line 16) | 3-10 |
| 3.3.8. | Special Display Control (Line 17) | 3-10 |
| 3.3.9. | Error Message Field (Line 19) | 3-11 |
| 3.3.10. | Display Retention (Line 20) | 3-11 |
| 3.3.11. | Function Keys (Line 21) | 3-11 |
| 3.3.12. | Nondisplayed Field (Line 23) | 3-12 |
| 3.4. | OPTIONAL SCREENS | 3-12 |
| 3.4.1. | Conditional Indicators | 3-13 |
| 3.4.2. | Conditional Erase/Replenish Screen | 3-15 |
| 3.4.3. | Edit Screen | 3-18 |
| 3.4.4. | Special Display Screen | 3-20 |
| 3.4.5. | Error Message Screen | 3-21 |
| 3.4.6. | Display Retention Screen | 3-24 |
| 3.4.7. | Function/Command Key Screen | 3-26 |
| 3.4.8. | Nondisplay Screen | 3-27 |
| 3.5. | DIALOG SCREENS | 3-28 |
| 3.5.1. | Dialog Screen 1 (General Field Characteristics) | 3-30 |
| 3.5.2. | Dialog Screen 2 (Input Field Response, Default, and Replenishing) | 3-35 |
| 3.5.3. | Dialog Screen 3 (Conditional Field Display) | 3-38 |
| 3.5.4. | Dialog Screen 4 (Special Field Display) | 3-40 |
| 3.5.5. | Dialog Screen 5 (Conditional Field Retention) | 3-43 |
| 3.5.6. | Dialog Screen 6 (Conditional Field Protection) | 3-44 |
| 3.5.7. | Dialog Screen 7 (Field Change Notification) | 3-45 |
| 3.5.8. | Dialog Screen 8 (Input Field Range Checking) | 3-46 |
| 3.5.9. | Default Values for Dialog Screens | 3-48 |
| 3.6. | COMPLETING SFG SCREENS – CURSOR POSITIONING AND OVERWRITING | 3-49 |
| 3.7. | ERROR AND WARNING MESSAGES | 3-51 |
| 3.8. | THE HELP FUNCTION | 3-53 |

4. CREATING SCREEN FORMATS

| | | |
|-------------|--|------|
| 4.1. | CREATING A SIMPLE SCREEN FORMAT | 4-1 |
| 4.1.1. | Pass 1 – Providing the Layout | 4-4 |
| 4.1.2. | Pass 2 – Specifying Type Attributes and Editing | 4-6 |
| 4.1.3. | Pass 3 – Specifying I/O Directions and Initiating Dialogs | 4-9 |
| 4.2. | CREATING MORE COMPLEX FORMATS (USING OPTIONAL AND DIALOG SCREENS) | 4-12 |
| 4.2.1. | A Sample Format | 4-12 |
| 4.2.1.1. | The Sample Home Screen and Characteristics Screen | 4-15 |
| 4.2.1.2. | The Conditional Erase/Replenish Screen for MYFORMAT | 4-16 |
| 4.2.1.3. | The Error Message Screen for MYFORMAT | 4-17 |
| 4.2.1.4. | Completing Pass 1 for MYFORMAT | 4-18 |
| 4.2.1.5. | Completing Pass 2 for MYFORMAT | 4-18 |
| 4.2.1.6. | Completing Pass 3 for MYFORMAT | 4-19 |
| 4.2.1.7. | Completing Dialog Screens for MYFORMAT | 4-21 |
| 4.3. | OVERLAY FORMATS | 4-25 |
| 4.3.1. | The Need for an Overlay Format | 4-25 |
| 4.3.2. | Creation of an Overlay Format | 4-28 |
| 4.4. | CREATING A FORMAT WITH DUPLICATE LINES | 4-29 |

5. MODIFYING FORMATS

| | | |
|-------------|---|-----|
| 5.1. | CREATE-FROM AND MODIFY FUNCTIONS | 5-1 |
| 5.1.1. | Modify Screen | 5-2 |
| 5.1.1.1. | Modify Screen – Option 1 (CHANGE TEMPLATE) | 5-3 |
| 5.1.1.2. | Modify Screen – Option 2 (CHANGE TYPE) | 5-4 |
| 5.1.1.3. | Modify Screen – Option 3 (CHANGE I/O) | 5-4 |
| 5.1.1.4. | Modify Screen – Option 7 (CHANGE TEMPLATE ONLY – NO INTERNAL CHANGES) | 5-4 |
| 5.2. | MODIFICATION SUMMARY | 5-6 |

6. ADDITIONAL FUNCTIONS

| | | |
|-------------|---|------|
| 6.1. | DISPLAY FUNCTIONS | 6-1 |
| 6.1.1. | SHOW Function | 6-1 |
| 6.1.2. | LIST Function | 6-1 |
| 6.1.2.1. | Summary Screen | 6-3 |
| 6.1.2.2. | Special Editing Display | 6-5 |
| 6.1.2.3. | Display Control and Error Message Display | 6-5 |
| 6.1.2.4. | Function/Command Key Display | 6-6 |
| 6.1.2.5. | The Format Display | 6-7 |
| 6.1.2.6. | List Screen | 6-7 |
| 6.1.2.7. | Sublist Screen | 6-10 |
| 6.1.2.8. | Conditional Values Screen | 6-12 |
| 6.1.2.9. | Input Record Screen | 6-14 |
| 6.1.2.10. | Output Record Screen | 6-14 |
| 6.1.3. | SPOOL Function | 6-15 |
| 6.2. | DELETE AND TERMINATE FUNCTIONS | 6-15 |

7. ASSOCIATING FORMATS WITH A PROGRAM: USING THE SCREEN FORMAT COORDINATOR (SFC)

| | | |
|----------|--|------|
| 7.1. | PROGRAMMER RESPONSIBILITIES | 7-1 |
| 7.2. | JOB CONTROL | 7-1 |
| 7.3. | PROGRAM CONSIDERATIONS | 7-5 |
| 7.3.1. | RPG II | 7-5 |
| 7.3.2. | COBOL | 7-9 |
| 7.3.2.1. | Major Coding Considerations | 7-10 |
| 7.3.2.2. | Using Indicators | 7-18 |
| 7.3.2.3. | Additional Coding Considerations for Function Keys and Multivolume Workstations | 7-21 |
| 7.3.2.4. | Considerations for Workstations with Interprogram Communication | 7-26 |
| 7.3.3. | FORTRAN | 7-26 |
| 7.3.4. | BAL | 7-27 |
| 7.3.4.1. | Open a File (OPEN) | 7-28 |
| 7.3.4.2. | Close a File (CLOSE) | 7-29 |
| 7.3.4.3. | Retrieve a Record (DMINP) | 7-30 |
| 7.3.4.4. | Select a Screen Format (DMSEL,SCREEN) | 7-32 |
| 7.3.4.5. | Output a Record (DMOUT) | 7-34 |
| 7.3.5. | ESCORT | 7-35 |
| 7.3.6. | IMS Programming Considerations | 7-36 |
| 7.4. | CONSIDERATIONS FOR THE WORKSTATION OPERATOR | 7-38 |

APPENDIXES

A. JOB CONTROL CONSIDERATIONS

| | | |
|------|---------------------------------------|-----|
| A.1. | THE USE SFS STATEMENT | A-1 |
| A.2. | JOB CONTROL FOR MULTIPLE WORKSTATIONS | A-6 |

B. THE 12-LINE SCREEN DISPLAY

C. FUNCTION KEYS

D. ALPHABET TABLE

E. SFS SUPPORT OF DISPLAY ATTRIBUTES

F. SAMPLE IMS ACTION COBOL PROGRAM FOR SCREEN FORMAT SERVICES (JAMENU)

→ G. BATCH FORMAT GENERATOR

INDEX**USER COMMENT FORM****FIGURES**

| | | |
|-------|---|------|
| 1-1. | Screen Format Used to Supply Input Data | 1-2 |
| 1-2. | Screen Format Used to Display Output Data | 1-3 |
| 1-3. | Screen Format Used for Input and Output | 1-3 |
| 1-4. | The Function of the SFG and SFC | 1-5 |
| 3-1. | The Home Screen | 3-2 |
| 3-2. | The Characteristics Screen | 3-6 |
| 3-3. | Conditional Erase/Replenish Screen | 3-15 |
| 3-4. | Edit Screen | 3-18 |
| 3-5. | Special Display Screen | 3-20 |
| 3-6. | Error Message Screen | 3-22 |
| 3-7. | Display Retention Screen | 3-24 |
| 3-8. | Function/Command Key Screen | 3-26 |
| 3-9. | Nondisplay Screen | 3-27 |
| 3-10. | Dialog Screen 1 | 3-30 |
| 3-11. | Dialog Screen 2 | 3-36 |
| 3-12. | Dialog Screen 3 | 3-39 |
| 3-13. | Dialog Screen 4 | 3-40 |
| 3-14. | Dialog Screen 5 | 3-43 |
| 3-15. | Dialog Screen 6 | 3-44 |
| 3-16. | Dialog Screen 7 | 3-45 |
| 3-17. | Dialog Screen 8 | 3-46 |
| 4-1. | Typical Template (Pass 1) Screen | 4-4 |
| 4-2. | Typical Type Attribute (Pass 2) Screen | 4-7 |
| 4-3. | Typical Initial I/O (Pass 3) Screen | 4-9 |
| 5-1. | Modify Screen | 5-3 |
| 6-1. | Summary Screen | 6-3 |
| 6-2. | Special Editing Display | 6-5 |
| 6-3. | Display Control and Error Message Display | 6-6 |
| 6-4. | Function/Command Key Display | 6-6 |
| 6-5. | Screen Format | 6-7 |
| 6-6. | List Screen | 6-7 |
| 6-7. | Sublist Screen | 6-11 |
| 6-8. | Conditional Values Screen | 6-13 |
| 6-9. | Input Record Screen | 6-14 |
| 6-10. | Output Record Screen | 6-15 |
| 7-1. | Screen Formats and Accompanying RPG II Specifications | 7-7 |
| 7-2. | Sample SCREEN01 | 7-12 |
| 7-3. | COBOL Program Entries for SCREEN01 | 7-14 |
| 7-4. | Sample SCREEN02 | 7-16 |
| 7-5. | COBOL Program Entries for SCREEN02 | 7-17 |
| 7-6. | COBOL Program Entries for Indicators | 7-20 |
| F-1. | Sample Action Program JAMENU Using Screen Formats | F-2 |

TABLES

| | | |
|------|--|------|
| 2-1. | Editing Rules for Signed Output Fields | 2-6 |
| 2-2. | Editing Examples | 2-7 |
| 3-1. | Characteristics Screen, Related Optional Screens, and Related Dialog Screens | 3-12 |
| 3-2. | Dialog Screens and Their Functions | 3-29 |
| 3-3. | Contents of Lines 16 and 17 of Dialog Screen 1 | 3-34 |
| 3-4. | Intensity/Emphasis Selections | 3-42 |
| 4-1. | Basic CREATE Function Steps | 4-1 |
| 4-2. | CREATE Function Steps Using Optional and Dialog Screens | 4-14 |
| 7-1. | Status Keys | 7-22 |
| 7-2. | Effects of Connect-Free Reporting | 7-24 |
| 7-3. | Effects of Function Key Reporting | 7-25 |

1. Introduction

1.1. PROVIDING INFORMATION VIA FORMS

One of the most common methods of providing and receiving information is through the use of forms. Whether we are filling them out, sending them in, or receiving them, almost every facet of activity, from recreation to business, involves the use of forms at some point. Applying for loans, credit cards, a driver's license; subscribing to magazines; preparing tax returns; even writing a check: doing any of these things necessitates handling a form.

Naturally, we take forms for granted and are even annoyed with the frequency at which we are required to handle them. We realize, however, that they provide one of the easiest and most effective ways for information to be supplied as well as received. They ensure a certain amount of uniformity that is essential whenever a transfer of information from one point to another is required.

How does this concern you? Well, since forms are capable of eliminating wasted time and confusion in other environments by standardizing the flow of information, they can provide a similar service in the data processing environment too – specifically by taking the complexity out of input and output methods. For this purpose, Sperry has designed screen formatting facilities.

1.2. PROVIDING INFORMATION VIA SCREEN FORMAT SERVICES

Screen format services is an interactive component of OS/3 that enables you to create formats (forms), not on paper, but on a video terminal – your workstation. Additionally, it provides you with the capabilities for easily maintaining these formats.

Once created, a format can be used:

- To input all the necessary data to a program
- To display output data from a program
- To supply input and display output

Inputting and outputting data via a screen format is extremely fast and simple. It is done on a fill-in-the-blanks basis. Practically anyone can input data to a program in this manner merely by keying in the information that the format requests. For output, the blanks of a screen format are automatically filled in with the proper information. Such output is easier to read as well as to interpret because it is formatted.

Figure 1-1 is an example of a screen format used for input purposes.

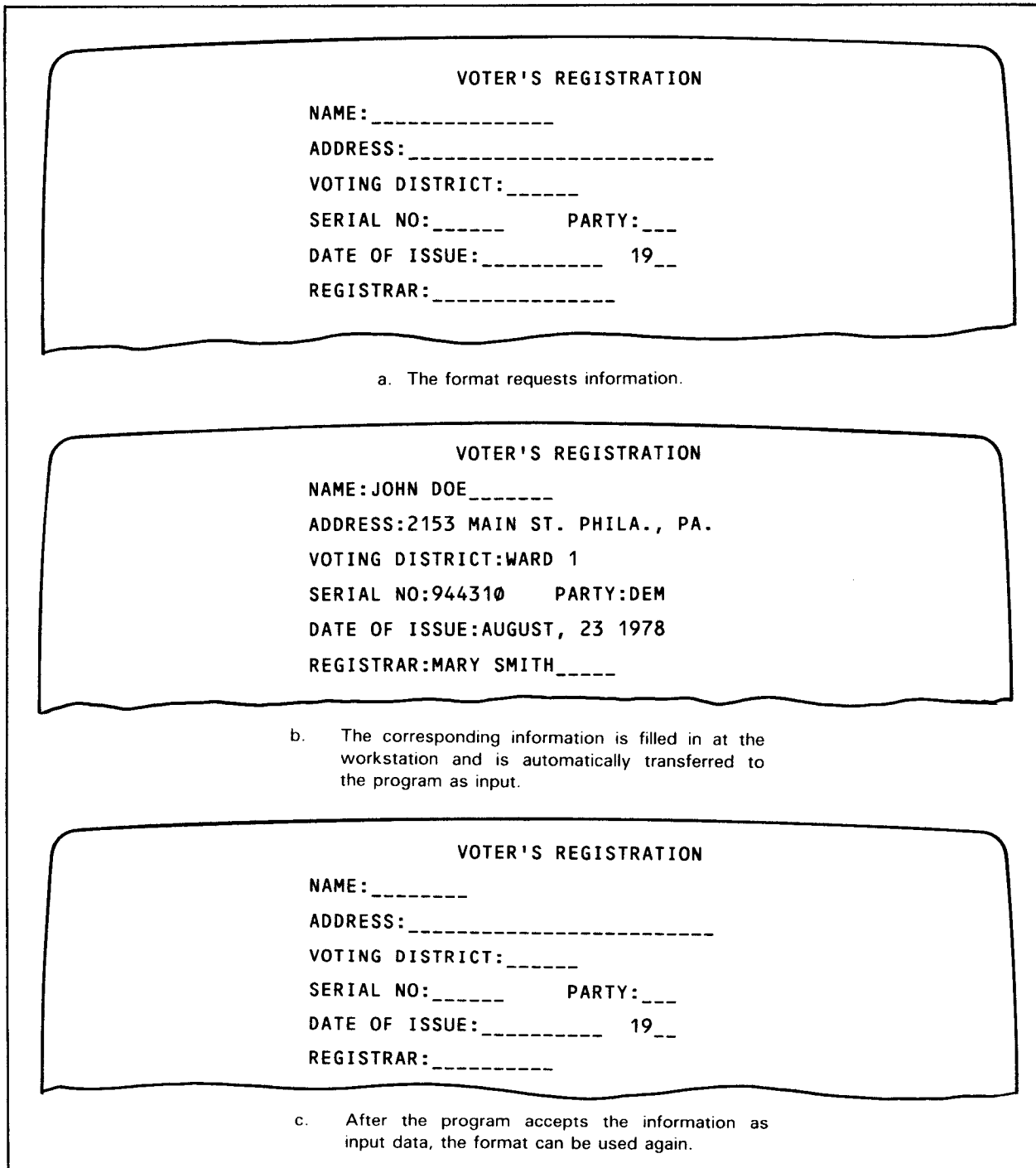


Figure 1-1. Screen Format Used to Supply Input Data

Figure 1-2 illustrates a screen format used for output purposes.

ACCOUNT STATUS REPORT

| | |
|---------------------|-----------------|
| ACCOUNT NO: 1827841 | DATE: 01/03/82 |
| CREDIT: 375.00 | |
| DEFICIT: 250.00 | BALANCE: 125.00 |

a. The screen format displays output.

ACCOUNT STATUS REPORT

| | |
|---------------------|-----------------|
| ACCOUNT NO: 1922640 | DATE: 01/03/82 |
| CREDIT: 250.00 | |
| DEFICIT: 100.00 | BALANCE: 150.00 |

b. The format can be used to display output of this type as long as necessary.

Figure 1-2. Screen Format Used to Display Output Data

Figure 1-3 illustrates how a screen format can serve both input and output purposes.

PERSONAL CREDIT REPORT

| | | |
|----------------------|----------------|------------|
| NAME: JOHN DOE | | |
| ADDR: 1552 MAIN ST. | STATE: PA | ZIP: 19140 |
| ACCOUNT NO: 193-A564 | | |
| BALANCE: 350.00 | | |
| PAYMENT: _____ | DATE: __/__/__ | |

a. The format displays information for a particular customer and requests information as well.

PERSONAL CREDIT REPORT

| | | |
|----------------------|----------------|------------|
| NAME: JOHN DOE | | |
| ADDR: 224 PINE ST. | STATE: PA | ZIP: 19102 |
| ACCOUNT NO: 193-A564 | | |
| BALANCE: 350.00 | | |
| PAYMENT: 25.00 | DATE: 12/23/82 | |

b. The workstation operator changes the customer's address and makes the appropriate entries for PAYMENT and DATE. Information on the format is passed to the program.

Figure 1-3. Screen Format Used for Input and Output (Part 1 of 2)

```
PERSONAL CREDIT REPORT
NAME:MARY SMITH
ADDR:1204 MASON AVE.      STATE:PA      ZIP:19054
ACCOUNT NO:193-B643
BALANCE:225.00
PAYMENT:_____      DATE:___/___/___
```

c. The format for another customer is then displayed at the workstation.

Figure 1-3. Screen Format Used for Input and Output (Part 2 of 2)

You create and use screen formats through the two software elements that make up screen format services: the screen format generator and the screen format coordinator.

The screen format generator (SFG) is the software element that allows you to generate screen formats at the workstation. During generation, you specify the layout of each format as well as other characteristics that best enable the format to meet the input and output requirements of a particular program. The SFG automatically stores the completed formats (as an FC-type module) for you either on a system library (\$Y\$FMT) that resides on the SYSRES volume or on a MIRAM user library on disk or format-label diskette that you specify when you create the formats. The SFG makes these formats readily available for you to modify, delete, or simply look at whenever necessary.

The screen format coordinator (SFC) becomes active when you actually want to use an existing format with a program.

The SFC automatically gets the format you request and displays it on the workstation screen so that filling in the blanks can begin. The SFC is, as its name implies, responsible for coordinating the transfer of input and output data between your program and the format that's displayed on the workstation screen.

Figure 1-4 illustrates how the SFG and SFC process your formats.

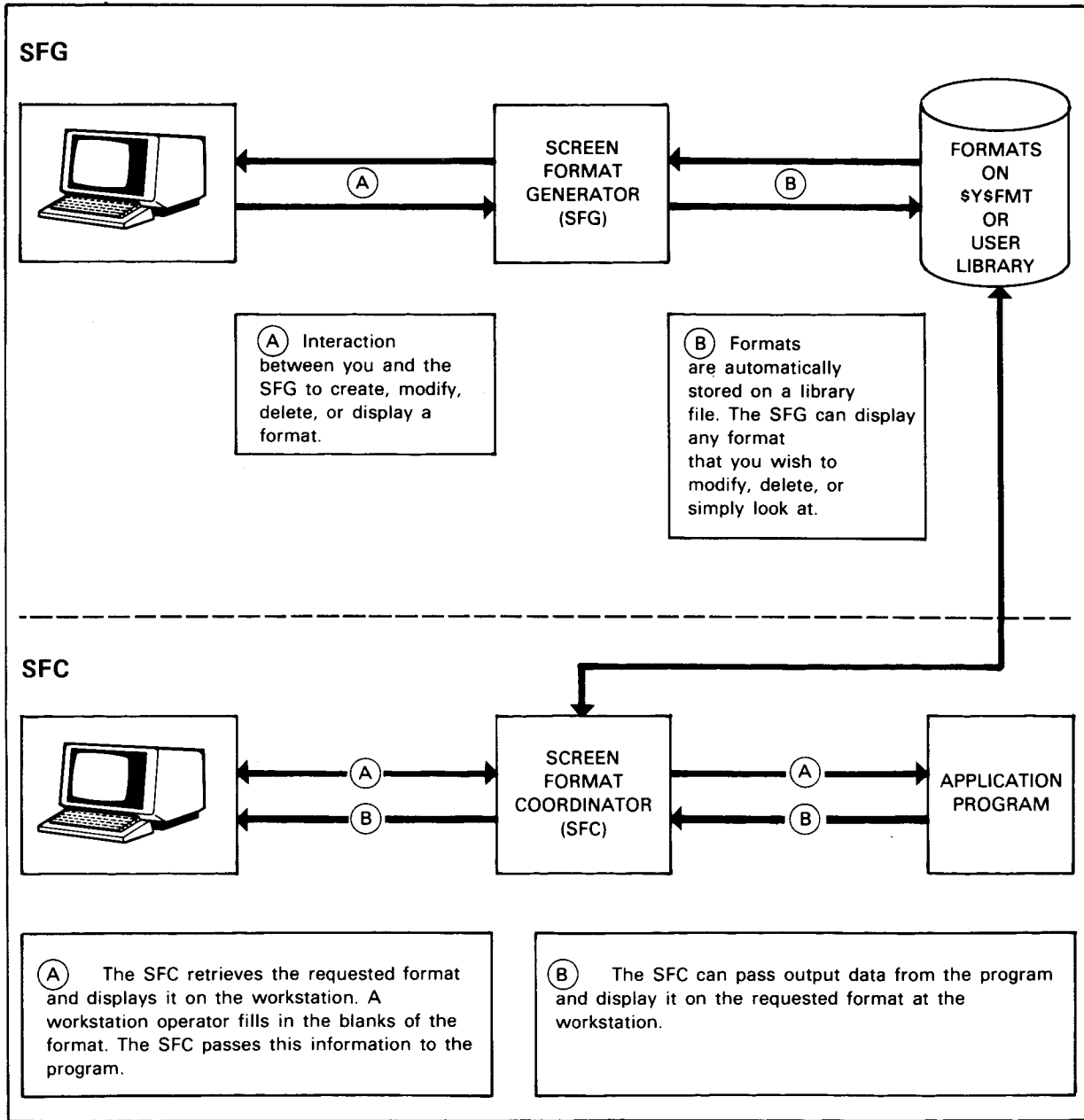


Figure 1-4. The Function of the SFG and SFC

Together, the SFG and the SFC provide you with everything you need to create, maintain, and ultimately use screen formats.

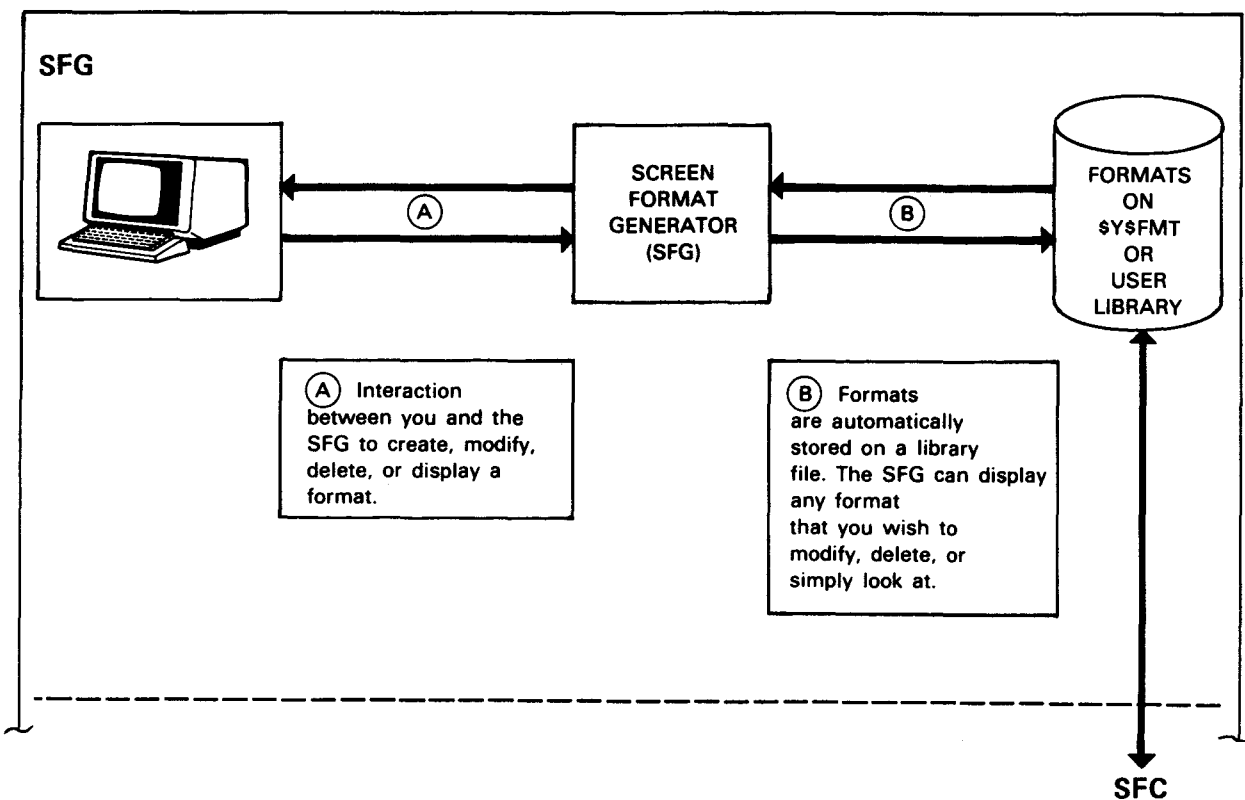
NOTE:

To be used successfully for screen format services, remote communications terminals (the UTS 200 or UTS 400) must have the field protect feature. If this feature is not provided, an error message (SF16) will occur whenever the operator enters data.

1.3. SCREEN FORMAT GENERATOR

1.3.1. Functions of the SFG

Let's concentrate on the functions of the SFG:



The SFG lets you create a format, modify (change) a format, display a format, and delete a format. With the ease of use characteristic of the SFG, practically anyone can perform these tasks - there's no need to be an experienced computer programmer.

You begin a format's creation (generation), for example, by answering a series of questions, collectively called the home screen, that are displayed at the workstation (see 3.2). You then key in the layout for the desired format. An example of such a layout is:

```
          PERSONAL CREDIT REPORT   TYPE:_  DATE:___/___/___  
NAME: _____  
ADDR: _____ STATE:___  ZIP: _____  
ACCOUNT NO: ___-_____  
PAST DUE AMOUNT: _____  
NEW BALANCE: _____ PAYMENT DUE DATE:___/___/___  
MINIMUM PAYMENT: _____
```

There are several steps, or passes, in which you can provide more specific information about the format, including whether it will be used for input, output, or both. These passes (explained in Section 4) are easy to complete and are generally the same for every format, so that after you've created one format, subsequent creations and modifications become almost second nature.

Other SFG functions are even simpler to complete. Before changing a format, for example, you might want to examine it to decide exactly what changes need to be made. In such cases, the SFG can automatically provide you with a display of any format stored on the library file. You need only supply the format's name. The format's name is likewise the criterion for deleting any format from the library. You supply the name, and the SFG does the work.

The SFG has additional features to make any of the format-related functions, particularly creation and modification, even easier to complete.

■ Error Indication

The SFG automatically notifies you of a variety of mistakes as soon as they are made. In most cases, the error is blinked on the workstation screen. In some instances, a specific error message is displayed. Thus, you are made aware of and can correct certain errors as soon as they occur.

■ Prompting

If you're not quite sure how to begin creating or modifying a format or if you're having difficulty along the way, you can ask the SFG for help simply by pressing a particular workstation function key (F13). The SFG then displays the instructions necessary to complete the operation causing the difficulty. You can be prompted in this fashion through as much or as little of any function (creating a format, modifying a format, etc) as necessary.

1.3.2. Screen Format Components

You now know that the SFG enables you to create, modify, delete, or display a format with a minimum of difficulty. Before we begin a more detailed discussion of these functions, however, you should know a little more about the nature of the format itself.

Formats can vary in size and complexity, as shown in the following screens.

```
          SALES REPORT
PRODUCT: _____
SALESMAN: _____
TOTAL SALES: _____
DATE: _____
```

```
          CLASS RANK
NAME: _____
GRADE: _____
GRADE POINT AVERAGE: _____
CLASS RANK: _____
```

```
          WAREHOUSE INVENTORY CONTROL
PROD. DESC: _____
STOCK NO: _____
MFR: _____
QTY ON HAND: _____
QTY ON ORDER: _____
REORDER QTY: _____
QTY DISCOUNT: __ % FOR QTYS OVER: _____
```

No matter how simple or complex, a format consists of only two components: display constants and variable data fields.

1.3.2.1. Display Constants

Display constants are portions of a screen format that never vary. For the most part, they serve as either headings or titles to introduce the information that will eventually fill in the blanks that follow. The shaded portions of the following format are display constants.

```
          SALES REPORT
    PRODUCT:-----
    SALESMAN:-----
    TOTAL SALES:-----
    DATE:-----
```

Once a format is displayed on the workstation screen in the first step of its creation, you cannot alter display constants in any of the successive steps. To change a display constant, you must initiate one of the SFG functions for modifying the format after it's created.

Likewise, when a format is being used to input data to or output data from a program, a display constant cannot accidentally be changed or blanked out either by the workstation operator or by any screen activity that results from the program. The program can never reference a display constant.

Other characters on a format, such as the slashes (/) separating the month, day, and year in a date, are also display constants. As a general rule, any character, other than an underline, that you enter when you first lay out the format is considered a display constant.

1.3.2.2. Variable Data Fields

Those portions of a format that are used either to input data to or to output data from a program are variable data fields. When creating a format, you always indicate variable data fields by underlines, with the number of underlines representing the external length of the field.

In successive steps during a format's creation, you make specifications as to the type of variable data a field receives (numeric, alphabetic, etc). Once the format is laid out, the remainder of the creation process deals exclusively with specifying everything from general to detailed characteristics for each variable data field in the format. The field's input and output directions are also specified, and since this (inputting and outputting) is the purpose behind generating formats in the first place, we'll discuss I/O directions briefly here.

Every variable data field must belong to one of three I/O classifications: input, output, or both input and output (bidirectional).

■ Input

Those fields designated during creation as input fields can only be used by the workstation operator to enter input data to a program at run time.

■ Output

Those fields designated during creation as output fields can only be used to display output data from the system at run time.

■ Both

Those fields designated during creation as bidirectional can be used either to input data to the program or to display output data from the system at run time.

The first time a format is displayed (when a program selects the format), input fields are (usually) represented with underlines. For example:

NAME: _____

Output fields, on the other hand, always display data.

NAME: MARY SMITH

An output field will never be displayed as

NAME: _____

unless the program supplies that field with underline characters.

A bidirectional field is always expected to receive output the first time a format is displayed. In other words, it acts initially as an output field.

NAME: MARY SMITH

Depending on the program specifications, a bidirectional field can continue to display output, or it can accept input, or it can do both. In the preceding field (NAME), the workstation operator could, for example, change the last name simply by entering a new name over it.

NAME: MARY JONES

The program accepts the new name as input.

The format shown in Figure 1-1 contains only input fields. The fields in Figure 1-2 could be either output fields displaying output or bidirectional fields displaying output. In Figure 1-3a, the fields NAME, STATE, ZIP, ACCOUNT NO, and BALANCE could be either output or bidirectional fields. You can be sure, however, that ADDR is a bidirectional field since the address is changed in Figure 1-3b. The fields PAYMENT and DATE are strictly input fields.

The procedure for specifying I/O directions for variable data fields is discussed in 3.5.1.

NOTE:

Any reference to fields in the remainder of this manual refers to variable data fields only.

1.3.2.3. Protected and Unprotected Parts – Low-Intensity Display

As you'll recall from 1.3.2.1, once defined, display constants cannot be altered either while successive passes of the creation process are being performed or while the completed format is being used with a program. These and other parts of a screen format with similar characteristics can be described as protected.

Depending on the particular part, there are differences in the extent to which it is protected. Also, in most cases the SFG automatically protects a part simply by virtue of its characteristics. In some cases, however, you can specify the protection yourself.

- Display Constants

Display constants are automatically protected as soon as you finish laying out the format. They remain so throughout the rest of the format's creation and also throughout any subsequent use of the format with a program until the entire format display is erased. Display constants are therefore totally protected and can only be changed via one of the SFG's modification functions.

- Output Fields

As explained in 1.3.2.2, a field that is specified for output during a format's creation is used exclusively to display variable data from the system. Displayed output data cannot be changed or erased by the workstation operator. Only new output coming from the system can overwrite previously displayed output. Naturally, output will be erased whenever the entire screen display is erased. It is in this sense that output fields are automatically protected.

- Specific Characters Used for Editing Input or Bidirectional Fields

At a certain point during creation of the format, you can specify editing for certain variable data fields by keying in COBOL-like picture clauses. If editing is specified for an input or bidirectional field, you have the option of specifying protection for certain characters in the picture string: the comma (,) and the decimal point (.). The SFG automatically protects a fixed currency symbol (\$). These characters become a sort of edit mask and, because they are protected, cannot be altered by a workstation operator or by the activity of the system.

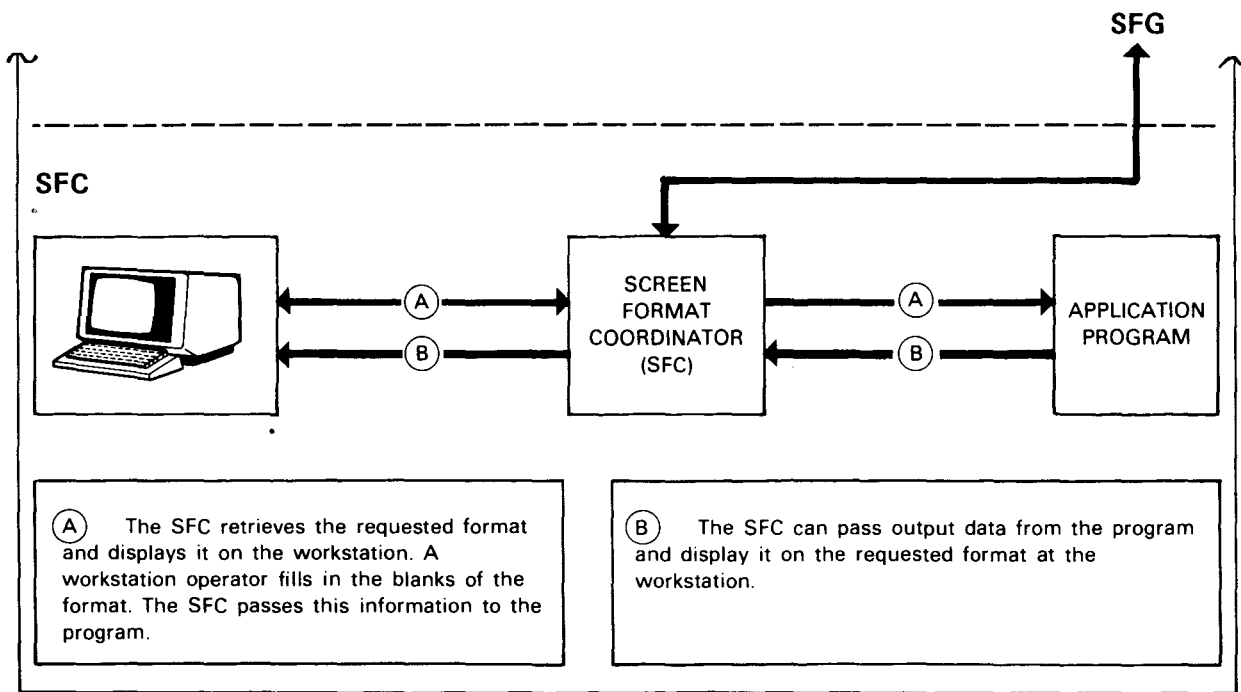
Although you should know what protection means, it isn't necessary for you to memorize which parts of a screen format are protected since the SFG makes the distinction. Protected parts are displayed on the workstation in low intensity. Low-intensity display simply means that the protected parts (characters) appear less bright than those that are unprotected.

NOTE:

During the creation or modification functions, there are several forms displayed by the SFG (the home screen, for example) the answers to which provide certain information about the format. Although not physically related to the format layout, portions of these displays are also protected and are displayed in low intensity.

1.4. SCREEN FORMAT COORDINATOR (SFC)

As mentioned earlier, the SFC becomes active when you want to use an existing format with a particular program for the purpose of inputting and outputting.



The SFC is responsible for retrieving the format (F-type module in \$Y\$FMT or your user library) you requested and displaying it on the workstation screen. Since the format is completely separate from your program, the SFC must also coordinate the flow of data between the program and the format.

In addition to transferring data back and forth, the SFC performs other necessary operations, such as checking or validating any input data that is filled in by the workstation operator. The data is checked against the specifications that you made for the field when you created or modified the format.

As will be explained in Section 4, there is a specific pass, or step, in the format's creation (for example, pass 2 - type attribute screen) where you indicate whether the field is to receive numeric, alphanumeric, or alphabetic data. If a field does not receive this type of data, the SFC will not accept the data as input to the program. Incorrect input is blinked on the workstation screen.

Since the SFC performs these and similar operations automatically, there's no need for you to be concerned with its activities at any time during the format's use. Simply be aware that the SFC handles the format according to the specifications you made during creation or modification via the SFG.

Although the SFC assumes the bulk of the responsibility for handling program input and output, you are responsible for such things as the proper job control statements associated with screen format services and for identifying as well as calling that format within the program. Specifics concerning job control statements and the necessary program statements as they relate to the particular language being used are discussed in Section 7.



2. Type Attributes and Editing Rules

2.1. CONSIDERATIONS FOR VARIABLE DATA FIELDS

As mentioned in 1.3.2.2, when you create a format, you specify variable data fields to handle the program's input and output data. In a specific step during the format's creation, you must define the type of data that each field is to receive. Data for variable fields can be in one of the following types: alphabetic, numeric, or alphanumeric. These are referred to as field type attributes.

With the SFG, you have the option of specifying editing for numeric fields and, to some extent, for alphabetic and alphanumeric fields as well. Since editing can be specified during creation of the format, your program has only to indicate such things as assumed decimal places (v) and signs (s) whenever appropriate, but not edit masks.

In Section 4, we'll discuss specifying type attributes and editing characters when creating a screen format. However, knowing the rules for type attributes and editing beforehand can be helpful.

2.2. TYPE ATTRIBUTES

The rules for indicating field types are similar to the rules applying to COBOL picture clause expressions:

- Alphabetic Fields

You use one or more A's to indicate that a particular field will receive alphabetic data when the format is in use. However, duplication factors (as A(2) in COBOL to indicate an alphabetic field that is two characters in length) are not supported.

A field defined as alphabetic can, at run time, contain any combination of the 26 characters of the alphabet and the space. These characters will be left-justified with blank padding.

■ Numeric Fields

You use one or more 9s to indicate that a format field can receive numeric data. Duplication factors are not supported. A field defined as numeric can, at run time, contain any combination of the numeric characters 0 through 9. A field of this type will be right-justified with zero padding.

■ Alphanumeric Fields

You can use any combination of Xs, As, or 9s to represent an alphanumeric field. Duplication factors are not supported. The positions indicated by Xs can, at run time, contain any combination of EBCDIC characters. Positions indicated by 9s or As can only contain the numbers 0 through 9 and alphabetic characters respectively. A field defined as alphanumeric will be left-justified with blank padding.

2.3. EDITING

Once you decide whether a field is to be alphabetic, alphanumeric, or numeric, your next consideration is whether you want to edit the field. Editing can be specified for any field in the format, although in some cases certain types of editing and edit characters restrict the field to handling output data only.

2.3.1. Editing Alphabetic and Alphanumeric Fields

You may use an exclamation point (!) to edit alphabetic and alphanumeric fields as shown in the following examples.

```
AAA!AAA  
XX!XX!XX  
XX!99!AA
```

This character reserves the position where it appears so that an insertion character that you select (such as a dash or a slash) can replace it. Positions indicated by the exclamation point are always protected and are never counted in the field's internal length. A field edited in this manner may be used for input, output, or both.

You may use the characters O (zero) and B in an alphanumeric field as shown in the following examples.

```
XXBXXOXX  
99BBXOAA
```

These characters reserve the positions where the numeral 0 and the space will be automatically inserted at run time. An alphanumeric field using either of these two characters may not use the exclamation point and vice versa. Also, such a field may be used only to display output data. The O and the B are counted in the external but not the internal field length.

NOTE:

External field length refers to the appearance of the field on the workstation screen, whereas internal length refers to the length of the field when referenced by the user program.

2.3.2. Editing Numeric Fields

You can edit a numeric field by using an exclamation point (!), insertion characters, or suppression.

The exclamation point gives you the same capabilities for numeric fields as it does for alphabetic and alphanumeric fields. A numeric field using the exclamation point may not, however, use any other form of editing and vice versa. Other SFG editing rules are similar in most aspects to COBOL editing rules.

The valid characters you can use for insertion or suppression editing are: asterisk (*), Z, plus sign (+), minus sign (-), CR, DB, decimal point (.), comma (,), blank (B), zero (0), and currency sign (\$).

You have the capability to move underlines (_) to numeric I/O fields on the screen. This will enable the operator to enter data directly without clearing the field of zeroes. User programs will not receive SF11 (USER IMPERATIVE ABNORMALLY TERMINATED) errors if the original data displayed contains underlines.

■ Insertion Editing

There are three types of insertion editing that you can specify during format generation: simple, fixed, and floating.

- Simple Insertion

You may insert the comma (,), the blank (B), and zero (0) into any numeric field as shown in the following examples.

```
9,999
9BB99900
```

The comma may be used in an input, output, or bidirectional field. Fields using the B or 0 are always output fields.

- Fixed Insertion

You may insert the currency sign (\$), plus sign (+), minus sign (-), credit symbol (CR), debit symbol (DB), comma (,), and decimal point (.) into any numeric field as shown in the following examples.

```
$9,999.99
+9.99
9,999.99CR
```

Fields using CR or DB are always output fields.

- Floating Insertion

You may express the currency sign (\$), plus sign (+), and minus sign (-) as floating signs as shown in the following examples.

```
$$$9.99
+999
```

The sign represents the leftmost position (before the first significant digit) into which it (\$, +, or -) will be inserted. A field edited in this manner may not use the CR or DB symbols and may only be used for output.

- Suppression/Replacement Editing

You may do this type of editing in one of the following ways:

- The character Z represents the suppression of leading zeroes (up to the first significant digit) for output only fields and their replacement with blanks as shown in the following examples.

```
ZZ,ZZ9
$Z,ZZ9.99
```

- The Z represents the suppression of leading zeroes (up to the first significant digit) for bidirectional fields and their replacement with the _ (e.g., _ _ _ _ 1.23). This allows you to see where any additional digits are entered.
- The asterisk (*) represents the suppression of leading zeroes (up to the first significant digit) for output only fields and their replacement with the * (e.g., ** 9.99).

An edited field cannot use the CR or DB symbols and can only be used for output. To use zero suppression on a bidirectional field, do not protect the punctuation because zeroes appear instead of underlines.

Refer to the 1974 ANSI COBOL programmer reference for more information concerning insertion and suppression/replacement editing for numeric fields.

2.4. RESULTS OF EDITING FORMAT FIELDS

You define type attributes and editing for variable data fields in pass 2 of the format's generation (see 4.1.2). In a later step (dialog screen 1, 3.5.1), you specify the characters to replace any exclamation points in a field, and optionally you can specify protection for certain characters in edited numeric fields (that are input or bidirectional), thus creating an edit mask.

Editing affects the display of a field when the format is presented at the workstation and consequently the way in which data can be entered (for input and bidirectional fields) by the workstation operator. Editing also affects the field's internal length, and in some cases it restricts a field to handling output data only.

How editing determines the workstation display and operator entries is especially obvious with edited numeric fields that are used for input.

Assume that during the format's creation a numeric input field is described as follows:

\$9,999.99

Since the SFG automatically protects a fixed currency sign, the field will be displayed as follows when the format is presented at the workstation:

\$_____

As long as the program indicates the number of decimal positions (two in this case), the workstation operator can make entries similar to any one of the following:

\$1,234.56

\$1.23_____

\$1234.56_

\$1234_____

Decimal point alignment is done automatically (by the SFC). If no decimal point is entered (e.g., \$1234), it is assumed to be to the right of the rightmost digit (\$1234.). Errors are indicated by blinking if the workstation operator makes an incorrect entry, such as:

\$123456__

The entry is in error because only four digits to the left of the decimal point are permitted (as indicated by the editing specified during the format's creation and the number of decimal places provided for in the program). If the workstation operator enters too many digits to the right of the decimal point (e.g., \$123.456), the value is accepted but the last digit (6) is truncated.

Suppose you want this field to be displayed on the workstation as follows:

\$_,____.____

In this case, you perform the same editing (\$9,999.99), but in addition you specify protection for the comma and the decimal point. The workstation operator must then enter the value in its proper position according to the field display. If the value is 1.23, the operator enters:

\$0,001.23.

Now consider format fields that use the sign +, -, CR, or DB. If you describe an input field as +999 or -999 during format generation, it will appear as _ _ _ _ when the format is displayed at the workstation. As long as the program indicates a sign for this field (e.g., S999 as in COBOL), the workstation operator can enter any 3-digit value with or without a sign. For example:

+123

-123

123_

The value with no sign is assumed to be positive.

For output fields using a sign, the (COBOL) rules in Table 2-1 apply.

Table 2-1. Editing Rules for Signed Output Fields

| If the Field Is Defined with | And the Output Value Is | Then the Output Is Displayed on the Format with |
|------------------------------|-------------------------|---|
| + (+999) | Positive or zero | + (+ 123) |
| | Negative | - (-123) |
| - (-999) | Positive or zero | Space (123) |
| | Negative | - (-123) |
| CR (999CR) | Positive | Two spaces (123) |
| | Negative | CR (123CR) |
| DB (999DB) | Positive or zero | Two spaces (123) |
| | Negative | DB (123DB) |

Fields that use the exclamation point (!) for editing are displayed on the format with whatever character you choose to replace the exclamation point with. Assume that you define the following field during the format's generation (pass 2).

SOCIAL SECURITY:999!99!9999

Later on during generation, you select a dash (-) to replace the exclamation point. If the field is used for input, it will be displayed as follows:

SOCIAL SECURITY:___-___-___

The operator can make the appropriate entries. If the field is used for output, it displays the data that is passed from your program to the format. For example:

SOCIAL SECURITY: 195-44-8867

Remember that edit characters used for simple or fixed insertion (such as B, O, \$, +, -, CR, DB, !, and .) are never counted in the field's internal length. Also, fields containing any acceptable combination of the floating signs \$, +, or -; the insertion characters CR or DB; or the suppression/replacement characters * or Z may only be used to display output data.

Table 2-2 lists some editing examples and the effects of that editing on the format fields.

Table 2-2. Editing Examples

| Field Type | Type Attributes and Editing Specified | External Length | Internal Length* | Field Display |
|--------------|---------------------------------------|-----------------|------------------|---|
| Alphabetic | AAA | 3 | 3 | INPUT: _ _ _ OUTPUT: ABC |
| | AAA!AAA | 7 | 6 | INPUT: _ _ _ _ _ _ OUTPUT: ABC-DEF (if dash is selected for replacement) |
| Alphanumeric | XXX | 3 | 3 | INPUT: _ _ _ OUTPUT: # 1A |
| | XX!99!AA | 8 | 6 | INPUT: _ _ _ _ _ _ OUTPUT: # 1-23-AB (if dash is selected for replacement) |
| | XXB9900XX | 9 | 6 | OUTPUT; ONLY: A3 4500BC |
| Numeric | 999 | 3 | 3 | INPUT: _ _ _ OUTPUT: 123 |
| | 99!99!99 | 8 | 6 | INPUT: _ _ / _ _ / _ _ OUTPUT: 12/24/80 (if slash is selected for replacement) |
| | \$9,999.99 | 9 | 6 | INPUT: \$ _ _ _ _ _ _ _ _ _ or (if , and . are protected) INPUT: \$ _ , _ _ _ _ _ _ OUTPUT: \$1,234.56 |
| | ZZ9.99 | 6 | 5 | OUTPUT ONLY: 123.45 or 12.34 |
| | \$\$9.99 | 6 | 4 | OUTPUT ONLY: \$12.34 or \$ 1.23 |

*Internal lengths indicated are based on the internal usage of display.

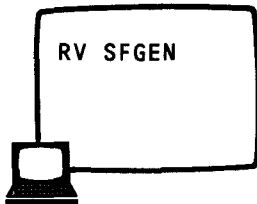


3. Using the Screen Format Generator (SFG)

3.1. ACTIVATING THE SFG

The first thing you must do to perform any of the SFG functions is activate the SFG. Do this on the workstation:

1. Log onto the system following the procedures outlined in the interactive services commands and facilities user guide/programmer reference, UP-9972 (current version).
2. Press the XMIT key.
3. Get into system mode by pressing the FUNCTION key and (while holding it down) pressing the SYS MODE key.
4. Key in RV SFGEN.



5. Press the XMIT key.

A formatted display called the home screen then appears on the workstation screen. The home screen is like a fill-in-the-blanks form: you enter the function you want to perform, as well as information about your screen format.

3.2. THE HOME SCREEN

The home screen (Figure 3-1) is always the first display presented whenever you activate the SFG. On it, you choose one of the eight SFG functions. If you're working on a format that already exists, you provide its name and library. If you're creating a new format, you name it, indicate where it will be stored, and, if necessary, allocate file space for it.

```

1.  Function      (1):  1 CREATE   2 CREATE-FROM  3 MODIFY   4 DELETE
2.                      5 SHOW     6 LIST       7 SPOOL    8 TERMINATE
3.
4.  Old Format:      Format name: (_____)
5.  is in library:  File name: ($Y$FMT )
6.                      Volume:   (RES  )
7.
8.
9.  New format:     Format name: (_____)
10. stored in library: File name: ($Y$FMT )
11.                      Volume:   (RES  )
12. File does not exist: Allocate:  (002) cylinders
13.                      Increment: (01)  cylinders
14.
15.
16.
17.
18. *Function keys are: F1 - GO TO HOME SCREEN  F5 - BREAKPOINT SPOOL FILE
19.                      F13 - HELP                F14 - EXIT HELP
20.                      F20 - RESTORE SCREEN
21.
22.
23.
24.

```

Figure 3-1. The Home Screen

Subsection 3.6 discusses how to position the cursor and how to overwrite values on the screen.

3.2.1. Functions Provided by the SFG

On line 1 of the home screen, you specify one of the eight functions provided by the SFG:

| | | | | |
|---------------|----------|---------------|----------|-------------|
| FUNCTION (1): | 1 CREATE | 2 CREATE-FROM | 3 MODIFY | 4 DELETE |
| | 5 SHOW | 6 LIST | 7 SPOOL | 8 TERMINATE |

■ CREATE

Use this function to create a screen format. The first step is to complete and transmit the home screen. The SFG then displays another fill-in-the-blanks screen called the characteristics screen. This screen asks you for format-related information and offers several options. Some of these options have other screens associated with them. If you choose these particular options, their optional screens are also displayed.

Once you've provided all the necessary information about your format, a blank screen is displayed for you to lay out your screen format. You supply any display constants and indicate any variable fields and their lengths. In two subsequent passes, you assign type (editing) attributes and input/output usages for each field on the format.

You also have the option of providing more information for each field in the format by initiating dialogs for those fields. Once your format is complete and transmitted, the home screen is displayed for you to choose another function.

■ CREATE-FROM

Use this function to create a new format by modifying an existing format. You can change the format as a whole, or you can change specifications on some or all of the fields in the format. The SFG saves the old format.

■ MODIFY

Use this function basically as you use the CREATE-FROM function. The only difference is that the SFG doesn't save the old format. The new format replaces the old format.

■ DELETE

Use this function to delete an existing format.

■ SHOW

Use this function to display an existing format and its type attributes on the workstation screen.

- LIST

Use this function to display information for each field within an existing format. The display includes such information as the creation date, the location of the fields on the screen format (x,y coordinates), input/output usage of the fields, internal and external field lengths, range checking and replenishing values, and conditional indicators associated with the field.

- SPOOL

Use this function to get a printout of an existing screen format, along with the field information provided by the LIST function. Because all printers do not support lowercase letters, the SPOOL function provides an option to translate lowercase to uppercase. This message appears after you select the SPOOL option:

```

Would you like the constants to be translated to uppercase before
the format is spooled?
(1) 1 NO    2 YES

```

The parentheses indicate that value 1 is the default value for this question. If you accept the default, lowercase alphabetic characters are translated to uppercase.

- TERMINATE

Use this function to terminate the SFG session and return control to the system.

3.2.2. New Format Name and Library

Every screen format must have a name and library. When you're creating a format, name it and indicate where it will be stored via lines 9, 10, and 11 of the home screen:

```

New format:          Format name:(_____)
stored in Library:  File name: ($Y$FMT          )
                   Volume:  (RES      )

```

Line 9 asks for the format name. The name can be eight characters and can contain only the letters A to Z, the numerals 0 to 9, and the special characters " # \$ @ and ?.

Spaces are not allowed within the name.

Lines 10 and 11 ask for the library where the format will be stored. \$Y\$FMT is a system file for storing formats; it is the default for FILE NAME. \$Y\$FMT resides on the system resident volume (RES). Thus, RES is the default for VOLUME. If you choose the defaults, no keyin is required. You're ready to transmit the home screen. You can, however, store your format on an alternate MIRAM library. To do so, overwrite \$Y\$FMT and RES with the appropriate file name and volume name.

If the alternate file you specify doesn't exist (hasn't been allocated), allocate space for it on the volume. Line 12 asks about the file space you need:

```
File does not exist: Allocate: (002) cylinders
                    Increment: (01) cylinders
```

If you need more than two cylinders for your file, overwrite the 002 (the default) with the number you need. The same applies for INCREMENT, which is the number of extra cylinders added to your file each time the file fills up.

The CREATE function isn't the only function that requires a new format name. The CREATE-FROM function also needs a new name because you're creating a whole new format based on the characteristics of an old one. The old format itself does not change, nor does its name.

3.2.3. Old Format Name and Library

The functions that require the old format name and library are:

| | |
|-------------|-------|
| CREATE-FROM | SHOW |
| MODIFY | LIST |
| DELETE | SPOOL |

On lines 4, 5 and 6, enter the name and library of the format you want to work with. If the format is stored on \$Y\$FMT, no further keyin is required. Just transmit the home screen. If the format is stored on some other library, enter the file name and volume where the format is located and then transmit the home screen.

NOTES:

1. To get a listing of existing format names, issue the FSTATUS command. For an explanation of this command and accompanying parameters, refer to the interactive services concepts and facilities user guide/programmer reference, UP-9972 (current version).
2. The Screen Format Generator may not be used to copy formats from one file on a diskette to the same file or to another file on that same diskette. ←

3.3. THE CHARACTERISTICS SCREEN

When you're creating a new format, the next screen displayed after the home screen is the characteristics screen. This screen (Figure 3-2) asks for more information about your format.

```

1. SPECIFY THE GLOBAL CHARACTERISTICS FOR FORMAT xxxxxxxx:
2.
3. Error retry count:          (02)
4. Alphabet:                  (ENGLISH )
5.
6. Lower case translation?    (1):      1 YES  2 NO
7.
8. Screen format is          (1):      1 ORIGINAL  2 OVERLAY
9.
10. Screen erase/unlock option (1):      1 NONE      2 REPLENISH
11.                               3 ERASE      4 UNLOCK KEYBOARD
12.                               5 CONDITIONAL INDICATOR
13.
14. Transmit all-disregard cursor position? (1):      1 NO  2 YES
15.
16. Special editing characters? (1):      1 NO  2 YES
17. Special display control?   (1):      1 NO  2 YES
18.
19. Error message field to be defined? (1):      1 NO  2 YES
20. Display retention on all fields? (1):      1 NO  2 YES
21. Function command keys to be defined? (1):      1 NO  2 YES
22.
23. Format has a non-displayed constant? (1):      1 NO  2 YES
24.

```

Figure 3-2. The Characteristics Screen

Some selections on the characteristics screen prompt additional screens. These are called optional screens and are discussed in 3.4. Optional screens further define certain run-time characteristics of your format. They automatically appear in sequence after you complete and transmit the characteristics screen.

Note that the first line of the characteristics screen always contains the format name you supplied on the home screen:

```
GLOBAL CHARACTERISTICS FOR FORMAT xxxxxxxx:
```

where xxxxxxxx is the 8-character name of your format.

3.3.1. Error Retry Counts (Line 3)

Your answer to this question determines the number of times a workstation operator may attempt to correct any input that is entered incorrectly.

ERROR RETRY COUNT: (02)

The value 2 is the default value for the number of times correction may be attempted. However, you may specify any value up to 99 by overwriting the default. If you specify 0, the operator can't perform a retry.

At run time, incorrect input is blinked (or displayed with the display attributes you select for fields entered in error) to the workstation operator. If the operator can't correct the input and the value specified for the retry count is reached, then an error status is returned to your program. You may make provisions within your program for this situation.

3.3.2. Alphabet (Line 4)

If you're creating (CREATE or CREATE-FROM) or modifying (MODIFY) a format, line 4 lets you indicate whether input provided by a workstation operator at run time is in a language other than English:

ALPHABET: (ENGLISH_)

English is the default alphabet. Appendix D lists the other valid alphabets for which translation tables are available.

3.3.3. Lowercase Translation (Line 6)

Line 6 asks you to specify whether lowercase alphabetic characters, entered by the workstation operator for the format's variable data fields, will be translated to uppercase before being passed as input to your program.

LOWER CASE TRANSLATION? (1): 1 YES 2 NO

As indicated by the parentheses, the default value for the question is 1 (YES). If you accept the default, lowercase alphabetic characters provided as input by a workstation operator are translated to uppercase before they are passed to the program. If you select 2 (NO), lowercase alphabetic characters aren't translated to uppercase.

NOTE:

No lowercase translation may be performed on Katakana workstations; therefore, you must overwrite the default value with 2 (NO), specifying no lowercase translation.

3.3.4. Original and Overlay Formats (Line 8)

An original format is simply the average screen format whose purpose is inputting and outputting information to and from your program. An overlay format serves the same general purpose. However, it is created for use with an original format. Your response to this part of line 8 indicates whether the format you're creating or modifying is an overlay or an original.

```
SCREEN FORMAT IS (1):  1 ORIGINAL  2 OVERLAY
```

An overlay format is created separately and may begin with a start-of-entry character (SOE), even those beginning in position (1,1). During format generation passes 2 and 3, the first line is shifted to the left one character, and the SOE character disappears. The screen is generated as specified in pass 1 with the SOE character, and no attempt should be made to shift back the incorrect line.

When selected by your program, the overlay format completely replaces (overlays) a portion of, or acts as an addition to, the original format that is already displayed. At this time, only the overlay is used for inputting or outputting because it's the only format recognized. The original format remains unchanged and exists only for display purposes. Creation of an overlay format and specific examples of its use are discussed in 4.3.

3.3.5. Erasing and Unlocking Options (Lines 10, 11, and 12)

When your program first displays a format on a screen, output-only and bidirectional fields contain data from your program, and input fields are ready to be filled in. When a workstation operator completes the format and transmits it back to your program, the next thing that occurs depends on the erase/unlock option you specified when you created the format. Your choices are displayed on lines 10, 11 and 12 of the characteristics screen:

```
Screen erase/unlock option      (1):      1 NONE      2 REPLENISH
                                   3 ERASE      4 UNLOCK KEYBOARD
                                   5 CONDITIONAL INDICATOR
```

■ NONE

This is the default option. It indicates that when a workstation operator transmits a format, the SFC verifies the input data. When verification is completed, the screen format itself remains on the workstation screen and the keyboard remains locked until your program requests the display of the same or a new format. This option offers you the most flexibility, in that your format isn't redisplayed automatically. If you intend to use conditional indicators to control your format's run-time characteristics (discussed later in this section), specify either this option or the CONDITIONAL INDICATOR IN USER PROGRAM option.

■ REPLENISH SCREEN

This option indicates that when a workstation operator transmits a format the SFC verifies the input data. Then, if verification is successful, the SFC immediately replenishes the input-only fields with a stringed constant (normally underlines) without waiting for your program to actually accept the input. The keyboard is then unlocked. Thus, a workstation operator can begin filling out another screen of data without waiting for your program to process the previous input data. This option is designed specifically for data entry applications that simply gather information from one format with a minimum of input verification. The formats should contain only input fields and should rely only on the verification that the SFC can perform.

■ ERASE SCREEN

This option indicates that after your program displays a format and an operator transmits the contents back to your program, the format is erased and the keyboard is unlocked. This option is recommended for applications that only intermittently use screen formats. It is not recommended for continuous data entry applications because there is no looping back to redisplay a format.

■ UNLOCK KEYBOARD

This option indicates that when an operator transmits a format, the SFC verifies the data, leaves the keyboard unlocked, and immediately allows a workstation operator to enter new data. It doesn't wait for your program to accept the input before it allows an operator to enter the next screen of data. In this respect, it resembles the REPLENISH SCREEN option. The difference between the two is that UNLOCK KEYBOARD doesn't replenish input fields with a stringed constant (typically underlines). Instead, it leaves the screen display as is, where input fields contain the previously entered data. All a workstation operator has to do is overwrite the old contents of the field with new data. Like REPLENISH SCREEN, this option is recommended for data entry applications where formats contain only input fields that require a minimum of verification.

■ CONDITIONAL INDICATOR IN USER PROGRAM

This option lets you set conditional indicators that determine your format's status. Depending on the conditions you specify and test for in your program, the input fields are replenished, the entire format is redisplayed, or the whole format is erased. You supply conditional erase/replenish indicators on an optional screen called the conditional erase/replenish screen. This screen (as well as any other applicable optional screens) appears after you complete and transmit the characteristics screen. For an explanation of the conditional erase/replenish screen, see 3.4.2.

A user program may erase the screen format or replenish input values by issuing explicit statements (e.g., a BAL programmer would use consolidated data management macros). The keyboard may also be unlocked this way. NONE or UNLOCK KEYBOARD should be chosen in this case. See 7.3.4 and the consolidated data management macro language user guide/programmer reference for the data management macros that can be used for screen control.

NOTE:

Do not specify REPLENISH SCREEN, ERASE SCREEN, or UNLOCK KEYBOARD if you intend to use range checking for any field in your format.

3.3.6. Transmit All-Disregard Cursor Position (Line 14)

If YES is chosen, all input will be returned to the user program displayed on the screen, regardless of where the cursor is positioned.

Transmit all-disregard cursor position? (1): 1 NO 2 YES

3.3.7. Special Editing Characters (Line 16)

This line asks if you'll use special editing characters for numeric edited fields:

Special editing characters? (1): 1 NO 2 YES

If you specify YES, an optional screen called the edit screen appears after you've transmitted the characteristics screen. The edit screen is discussed in 3.4.3.

3.3.8. Special Display Control (Line 17)

Line 17 asks if you want to change the display properties of the different types of fields in your format:

Special display control? (1): 1 NO 2 YES

If you choose YES, you prompt the display of an optional screen called the special display screen (3.4.4). On the special display screen, you can indicate the intensity and emphasis for display constants, input and output fields, and fields where an operator has made an error. If you accept the default, NO, the default display properties are:

- Low intensity and no emphasis for display constants and output fields
- Normal intensity and no emphasis for input fields
- Normal intensity and blinking for fields in error

3.3.9. Error Message Field (Line 19)

You have the option of specifying where error messages from your program are displayed on your format, under what conditions they should appear, and how they look when they are displayed. To request an error message field, respond to line 19 of the characteristics screen:

Error message field to be defined? (1): 1 NO 2 YES

If you answer (2) YES, you prompt the display of the optional error message screen (3.4.5.)

3.3.10. Display Retention (Line 20)

Line 20 lets you request conditional display retention for your format. Conditional display retention means that, under certain circumstances, the contents of all variable data fields remain displayed on the screen after a workstation operator transmits the format. Input fields aren't replenished with underlines (or other stringed constant); instead, they retain the values the workstation operator keyed in. Similarly, output and bidirectional fields don't receive new values from your program. The only thing the SFC actually sends out to your format is the field display attributes (i.e., the intensity and emphasis). Conditional display retention has various uses that are discussed in detail in 3.4.6. To request display retention for your format, respond to line 20 of the characteristics screen:

Display retention on all fields? (1): 1 NO 2 YES

By responding with 2 (YES), you prompt the display of an optional screen called the display retention screen (3.4.6).

3.3.11. Function Keys (Line 21)

You can assign your own functions or commands to function keys 1 through 22 on the keyboard. At run time, an operator simply presses the appropriate keys to execute the function you've assigned to it in your program. By replying (2) YES to line 21 of the characteristics screen,

Function or command keys to be defined? (1): 1 NO 2 YES

you prompt the display of the function/command key screen (3.4.7), where you select the keys you want to use, along with their run-time indicators.

You have the capability to have one response indicator set by more than one function key. In fact, any combination of function keys and response indicators can be specified.

NOTE:

When executing SFG from the UNISCOPE 400, you may not request function key specification.

3.3.12. Nondisplayed Field (Line 23)

Each format is allowed one field that communicates with your program but never appears on the workstation screen. When you enter 2 (YES) on line 23 of the characteristics screen,

Format has a non-displayed constant? (1): 1 NO 2 YES

you prompt the display of the nondisplay screen (3.4.8), where you specify how the field will be used.

3.4. OPTIONAL SCREENS

There are seven optional screens associated with the characteristics screen. Each one gives you additional control over some run-time feature of your format.

The selections from the characteristics screen that prompt optional screens are listed in Table 3-1 and discussed in the following paragraphs. Note that whatever you specify on an optional screen affects every field in your format at run time. Some of the functions that optional screens provide, however, can be specified at the field level via a dialog screen. (Dialog screens are discussed in 3.5.) For example, you can specify display retention at the screen level as well as at the field level. However, at run time, your field level specification always overrides what you specify at the format level. For easy reference, the dialog screens that correspond to optional screens are also listed in Table 3-1.

Table 3-1. Characteristics Screen, Related Optional Screens, and Related Dialog Screens

| Selection from Characteristics Screen | Resulting Optional Screen | Corresponding Dialog Screen |
|---|--|-----------------------------|
| SCREEN ERASE/UNLOCK OPTION (5): CONDITIONAL INDICATOR IN USER PROGRAM | Conditional erase/ replenish screen | |
| SPECIAL EDITING CHARACTERS? (2): YES | Edit screen | Dialog screen 2 |
| SPECIAL DISPLAY CONTROL? (2): YES | Special display screen | Dialog screen 4 |
| ERROR MESSAGE FIELD TO BE DEFINED? (2): YES | Error message screen | |
| DISPLAY RETENTION ON ALL FIELDS? (2): YES | Display retention screen | Dialog screen 5 |
| FUNCTION COMMAND KEYS TO BE DEFINED? (2): YES | Function command/key screen | |
| FORMAT HAS A NON-DISPLAYED CONSTANT? (2): YES | Nondisplay screen | |

If you request any optional screens, they automatically appear, in the order discussed in the following subsections, after you transmit the characteristics screen. Once you complete and transmit the optional screens you requested, you're presented with a blank screen for laying out your format (pass 1, 4.1.1).

Some general rules to keep in mind when working on optional screens are:

1. If you change your mind about using an optional screen but the screen has already come up on your workstation, just transmit the screen without entering anything on it. The SFG then accepts the default values for that option.
2. You must correct all errors (indicated by blinking or message) before you can successfully transmit an optional screen. To ignore warning messages, position the cursor at the end of the screen and press the transmit key.
3. You can initiate the HELP function at any point.
4. Rules concerning the position of the cursor are the same for optional screens as for the home screen (3.6.)

Many optional screens, as well as many dialog screens, require conditional indicators. Before we discuss optional screens, it will help you to understand what these indicators do.

3.4.1. Conditional Indicators

Conditional indicators are a means of *conditionally* controlling certain run-time features of your format. In your program, you define a set of conditions under which you want a particular feature of your format (or a single field) to change. To each condition, you assign a number that serves as an indicator. You enter this indicator number on the optional screen or dialog screen associated with the feature you want to control. At run time, your program tests to see if that condition is true. If it is, it turns on the indicator. The indicator acts as a flag to the SFC, telling it to alter that feature of the format or field.

You can associate a single indicator with more than one optional or dialog screen. That is, one indicator number can control several different run-time features of your format. For example, a single indicator can cause your program to display an error message, highlight a field in error, and even protect the other fields that an operator entered correctly.

There are four ways of specifying how an indicator should work:

■ (Y nnn) syntax

This is how the indicator is normally used. The Y implies that some action is taken when a condition is true and your program turns an indicator on.

■ (N nnn) syntax

By overwriting the Y with an N (the Y is not protected), you specify that some action is taken only when an indicator is off (that is, only when the condition in your program is false).

■ (Y 000) syntax

By entering zeros instead of a number or by leaving the indicator field blank, you set up an unconditional indicator. It means that some action (like display retention) always applies. (In effect, the condition is always true.)

■ (N 000) syntax

This, too, is an unconditional indicator. It means that some action (again, like display retention) never applies. (The condition is always false.) You can achieve the same effect by leaving the field blank, rather than filling in zeros.

Use of the unconditional syntax is sometimes discouraged or redundant, depending on the application. Where this is true, it is noted in the following discussions of optional screens and dialog screens.

The fields for the indicator numbers that appear on optional and dialog screens are three character spaces long. If your indicator is less than three characters, you still enter the first number at the first character slot. For example, you enter indicators 1, 12, and 123 as follows:

(Y 1__) (Y 12_) (Y 123)

NOTE:

RPG II users are limited to indicator numbers 1 to 99.

When you specify a replenish indicator, this is what happens:

■ Step 1

Your program displays a format at a workstation screen. (The format can contain any combination of output, input, and bidirectional fields.)

■ Step 2

A workstation operator fills in and then transmits your format.

■ Step 3

The SFC accepts the data from the format and performs its own validation on the input. If successful, it then passes the input to your program. The keyboard remains locked.

■ Step 4

Your program performs its own validation of the input. If and when the input meets a condition your program is testing for, it decides that only the input-only fields should be replenished. Your program then turns on your replenish indicator and signals the SFC.

■ Step 5

The SFC replenishes your input-only fields with a stringed constant (like underlines) and unlocks the keyboard. *It doesn't replenish bidirectional and output fields.* That is, these fields don't receive new output from your program. They retain the values they had when the operator transmitted your format at step 2. The associated attributes of output and bidirectional fields are rewritten, however.

When the SFC replenishes the input fields and unlocks the keyboard, the workstation operator can enter new data. This time when the operator transmits the format, the complete record that the SFC accepts and sends back to your program consists of the new data the operator just supplied, as well as the data that was retained and unchanged in the bidirectional fields.

To enter a replenish indicator on the conditional erase/replenish screen, you must first answer the question on line 6 and then enter an indicator number:

Do you wish all input-only fields to be REPLENISHED after input based upon a conditional indicator?

(1): 1 NO 2 YES

INDICATOR VALUE: (Y ___)

The erase indicator you supply on line 11 of the conditional erase/replenish screen lets your program conditionally erase the entire screen format. The erase indicator acts as a signal to the SFC to completely erase, and thus end, the display of a particular format. Typically, you use an erase indicator to signal the end of a workstation session. Like the replenish indicator, your program decides when to turn an erase indicator on, based on the input it receives from a workstation operator.

The steps described for the replenish indicator apply also to the erase indicator through step 3. At that point, the following occurs:

■ Step 4

Your program validates the input. If it meets criteria, your program turns the erase indicator on and signals the SFC.

■ Step 5

The SFC, in response, erases the workstation screen and unlocks the keyboard.

To enter an erase indicator on the conditional erase/replenish screen, you must first answer the question on line 11 and then enter an indicator number:

Do you wish to have the screen ERASED after input based upon a conditional indicator?

(1): 1 NO 2 YES

INDICATOR VALUE: (Y ___)

So far, we've discussed what happens when either a replenish or an erase indicator is on. The other possibility, not yet covered, is what happens when neither indicator is on. When this is the case, your format remains under the normal control of your program (as it would be if you specified NONE as your erase/unlock option). The screen format remains on the workstation screen and the keyboard remains locked until your program requests the display of the same or a new format or, perhaps, turns on other indicators that affect your format's run-time characteristics.

For a sample format that uses the conditional erase/replenish option, see 4.2.

NOTE:

Your replenish and erase indicators take precedence over any other indicators you specify. All other indicators are ignored in this type of output operation.

3.4.3. Edit Screen

The edit screen (Figure 3-4) is displayed only if you elect, via line 16 of the characteristics screen (SPECIAL EDITING CHARACTERS? (1): 1 NO 2 YES), to replace certain numeric edit characters with those of your choice.

You specify a special edit character by overwriting the character that appears in parentheses. You could, for example, overwrite the currency sign (\$) with the pound symbol (#). You must then use the special edit character (#) instead of the currency sign when completing the type attribute screen in pass 2. Thus, a numeric edited field might be displayed on the completed format as #9,999.99 instead of \$9,999.99.

Using the edit screen, you may select a character other than the underline (_) to replenish all input and bidirectional fields. You can also do this on a field-by-field basis by using dialog screen 2 (3.5.2).

| | | | |
|-----|---|-----------------------|------------------------|
| 1. | ENTER EDIT CHARACTERS IN PARENTHESES TO BE USED FOR FORMAT xxxxxxxx: | | |
| 2. | | | |
| 3. | STANDARD CHARACTER : | MEANING: | = CHARACTER TO BE USED |
| 4. | | | |
| 5. | \$: | currency symbol | = (_) |
| 6. | . : | decimal point | = (_) |
| 7. | , : | thousands punctuation | = (_) |
| 8. | * : | replacement character | = (_) |
| 9. | CR: | credit on negative | = (_) |
| 10. | DB: | debit on negative | = (_) |
| 11. | _ : | replenish character | = (_) |
| 12. | | | |
| 13. | | | |
| 14. | NOTE: Make sure no symbol is ambiguous with any valid picture string character. | | |
| 15. | | | |

Figure 3-4. Edit Screen

NOTE:

For Katakana, the default currency sign on the edit screen is the yen (Y) instead of the dollar sign (\$).

Some general rules pertaining to the completion of the edit screen are:

1. Special edit characters must be the same length as the original.
2. With the exception of the replenishing character, you may not select any of the following as special edit characters: !, @, A, X, 0 (zero), B, E, +, - (minus), Z, L, H, or the underline.
3. Special edit characters must not be ambiguous and must be different from each other.
4. If you enter an unacceptable edit character, it will be blinked. The edit screen can't be transmitted until you correct all errors.
5. If you decide you don't want to select any special edit characters, simply transmit the edit screen as is.
6. You may request help or return to the home screen by using the appropriate function keys (F13 and F1).
7. If you want your program to receive the underline character () that an operator keys in to an input or bidirectional field, you must specify a blank as the replenish character. Otherwise, if you use the underline as your replenish character, the SFC automatically translates any underlines it detects to blanks. Thus, the underline the operator keys in will never make it back to your program.

Note that if you use blanks as your replenish character, the workstation operator can't see where input fields begin and end. (No stringed constant appears after a display constant.) If the workstations you use are model 2 (UTS 40D) terminals, you can get around this restriction. The model 2 workstations let you treat underlines as a display attribute that you specify on a field-by-field basis. See dialog screen 4 (3.5.4) for details.

3.4.4. Special Display Screen

By requesting special display control on the characteristics screen, you prompt the special display screen to appear as shown in Figure 3-5.

```

1.  ENTER THE DISPLAY PROPERTIES FOR FORMAT xxxxxxxx:
2.
3.                                     Intensity      Emphasis:
4.      Display constants                (B)          (____)
5.      Output variables                 (B)          (____)
6.      Input capable fields             (A)          (____)
7.      Fields entered in error          (A)          (5____)
8.
9.
10.
11.      Intensity:                       Emphasis:
12.      A. NORMAL INTENSITY              1. UNDERLINED      4. REVERSE VIDEO
13.      B. ALTERNATE (LOW) INTENSITY     2. COLUMN SEPARATORS 5. BLINK FIELD
14.      3. STRIKE THRU BARS
15.

```

Figure 3-5. Special Display Screen

The special display screen lets you select the display properties for the four types of fields in your format: display constants, output variables, input variables (those defined as input-only or bidirectional), and fields where an operator has made an error entering data.

When the special display screen first appears, it contains the SFG defaults for intensity and emphasis (the letters A and B and the numbers 1 through 5 refer to the key that appears at the bottom of the screen). The defaults are:

- Output fields: low intensity (B), no emphasis
- Input fields: normal intensity (A), no emphasis
- Display constants: low intensity (B), no emphasis
- Fields in error: normal intensity (A), blinking emphasis (5)

You can change one or all the SFG defaults by entering your own choices. For each field, you can specify one intensity and up to five different types of emphasis. For example, you can specify that fields in error be displayed in normal intensity along with blinking, underlining, and reverse video, all at the same time. You would simply fill in line 7 of the special display screen as follows:

| | INTENSITY | EMPHASIS |
|-------------------------|-----------|----------|
| FIELDS ENTERED IN ERROR | (A) | (145__) |

When specifying display combinations, keep in mind the operator who will be using your format. Some display combinations could result in eye strain.

The specifications you make on the special display screen become the new defaults for your format. As you'll see in 3.5.4, you can override the format specification for individual fields via dialog screen 4.

NOTE:

Not all display properties are supported by all devices. If the type of workstation where your format is being used doesn't support the display combinations you selected, the system selects alternate display properties at run time. Appendix E lists these alternate display properties.

3.4.5. Error Message Screen

If you want the ability to display an error message from your program on a workstation screen, you must designate a field for it on your format. Through the error message screen, you can specify where and when the error message is displayed, as well as how it should look. You prompt the error message screen by responding as follows to line 19 of the characteristics screen:

Error message field to be defined? (2): 1 NO 2 YES

Figure 3-6 is a display of the error message screen as it first appears.

```

1.  ENTER THE FOLLOWING INFORMATION FOR YOUR ERROR MESSAGES FOR FORMAT xxxxxxxx:
2.
3.  Message field name is:                (ERRORMSG)
4.
5.  Number of lines in error message:    (1)   ENTER 1 OR 2
6.  Error message to be displayed on line number: (LAST) ENTER NUMBER OR 'LAST'
7.
8.  Error messages conditionally displayed: indicator (Y ___)
9.
10. Display attributes:                   Intensity (A)
11.                                     Emphasis: (____)
12.
13.
14.
15.
16.      Intensity:                        Emphasis:
17.  A. NORMAL INTENSITY                   1. UNDERLINED           4. REVERSE VIDEO
18.  B. ALTERNATE (LOW) INTENSITY          2. COLUMN SEPARATORS  5. BLINK FIELD
19.                                       3. STRIKE THRU BARS
20.

```

Figure 3-6. Error Message Screen

The screen can be analyzed as follows:

■ Line 1

Contains the name of the format being generated, where xxxxxxxx is the 8-character name of your format.

■ Line 3

Displays the name of the error message field. The SFG assigns the name ERRORMSG to the error message field. If you prefer, you can enter your own, application-oriented name. It must be unique and must begin with an alphabetic character and can be a maximum of eight characters. It can contain any valid alphabetic character, the numerals 0 to 9, and the special characters \$, #, a, x, @, and ?. No embedded spaces are allowed.

■ Line 5

Asks you to designate one or two lines (60 or 120 characters) for your error message. The default is one line. Note that even if the text of your error message is less than 60 or 120 characters, you should assign either 60 or 120 characters for it in your program. This ensures that when your error message is displayed, the entire field is filled with either text or blanks. Otherwise, since the SFC expects either 60 or 120 characters, it could fill the field in with meaningless data from your work area.

- Line 6

Displays the location of the error message field. The error message field location defaults to the last line of the screen. You can, however, specify an actual line number (or numbers for 2-line messages) indicating where you want the error message to appear. You should be careful, however, not to choose a line that will contain input or bidirectional fields. If you inadvertently do so, the screen format generator won't detect it; at run time it could result in erroneous data being returned to your program, or you'll get an error message from the SFC.

- Line 8

Asks you for a conditional indicator. If your program detects an error when it validates input, it can turn on the error message indicator. The SFC, in response, displays the error message on your format. (You can also use the N nnn form for the indicator, where as long as the indicator is on, the error message is not displayed. When the indicator is off, it is displayed. Unconditional indicators aren't suitable for this feature.)

- Lines 10 and 11

Displays the default display properties (intensity and emphasis qualities) for the error message field. The defaults are normal intensity (A) and no emphasis. To specify your own choice of display properties, overwrite the default values, using the key at the bottom of the screen for reference. You can combine one intensity with as many as five types of emphasis.

- Lines 16 through 19

Display a key for entering your choice of display properties. The letters A and B refer to intensities, and the numbers 1 through 5 refer to types of emphasis.

NOTES:

1. *The error message field must be the last field you define in the output record area of your program.*
2. *The error message indicator overrides any other user-specified retain options.*

When the error message indicator is on, the display of the message isn't the only thing that occurs. The SFC also retains the contents of all variable data fields on the screen and rewrites the display attributes of each field. This gives you two important advantages.

First, your program typically displays an error message when an operator enters and transmits the wrong data. By retaining the contents of variable data fields, the operator can see exactly what he entered (and what caused your program to display the error message).

Secondly, by completely rewriting the field display properties (the intensity and emphasis of each field), the SFC can highlight the field where the operator made the error. This ability to highlight a field in error involves more than what the error message screen can provide. To highlight a field, you need to specify a conditional indicator that controls its display properties. You specify such an indicator via dialog screen 4 (3.5.4), the special display properties dialog.

3.4.6. Display Retention Screen

The display retention screen (Figure 3-7) lets you set an indicator that controls whether the contents of variable data fields are retained on the workstation screen.

```
1. Do you wish the display image to be CONDITIONALLY RETAINED
2. for all fields of format xxxxxxxx?
3.
4. (1): 1 NO 2 YES          INDICATOR VALUE: (Y ___)
5.
6.
7. NOTE: If this option is selected, an output command will affect display
8. attributes only, unless overridden at field level via DIALOG 5.
9.
10.
11.
```

Figure 3-7. Display Retention Screen

Display retention occurs when your program receives data from your format and, based on the nature of the data, turns the retain indicator on. In response to the retain indicator, the SFC doesn't change the text of the workstation screen. Output fields retain the values they previously received from your program; they aren't replenished with new values. Bidirectional fields retain either the value supplied by your program or the value a workstation operator keyed in over it. Input fields are left with the data the workstation operator keyed in; they aren't replenished with underlines or other stringed constant.

The only thing the SFC actually rewrites to the screen is each field's display attributes (that is, their intensity and emphasis). If the display attributes don't change, an operator won't notice a difference in the format. However, if the SFC sends out new display attributes to a field (such as blinking or reverse video), the operator will notice a very visible difference.

You can use the display retention indicator with other indicators. For example, if your program detects an error in the input it received from your operator, it can turn on two indicators: a retain indicator that preserves the data on the screen, thus allowing an operator to see what he entered, and an indicator that changes the display attributes of the field in error, such that the field stands out from the others. When the workstation operator corrects the field in error and transmits the screen, the SFC accepts the newly corrected value the operator entered, as well as the other values that were retained on the screen. (See dialog screen 4 (3.5.4) for an explanation of how to conditionally control field display attributes.)

You can also specify display retention at the field level via dialog screen 5. Note, however, that your field-level indicator always overrides the format-level indicator. This being the case, you can selectively retain all or just some of the fields on your format.

NOTE:

The first time a format is written to a workstation screen, the SFC ignores any retain indicators you specify.

To supply an indicator on the display retention screen, you must first answer the question on line 4:

```
DO YOU WISH THE DISPLAY IMAGE TO BE CONDITIONALLY RETAINED
FOR ALL FIELDS OF FORMAT xxxxxxxx?
  (2)  1 NO  2 YES          INDICATOR VALUE: (Y ___)
```

You then supply the indicator value, also on line 4. (If desired, you can use the unconditional Y 000 or Y _ _ _ forms of the indicator, as well as the N nnn form.)

3.4.7. Function/Command Key Screen

The function/command key screen (Figure 3-8) lets you designate which keys you assign your own program functions to. The allowable keys are function keys 1 through 22 on the workstation keyboard.

| 1. | ENTER THE FUNCTION/COMMAND KEYS TO BE USED BY FORMAT xxxxxxxx: | | | | | | | | |
|-----|--|---------|----------|------|---------|----------|------|---------|----------|
| 2. | | | | | | | | | |
| 3. | Key# | Accept? | Response | Key# | Accept? | Response | Key# | Accept? | Response |
| 4. | | | | | | | | | |
| 5. | F1 | (N) | (___) | F2 | (N) | (___) | F3 | (N) | (___) |
| 6. | F4 | (N) | (___) | F5 | (N) | (___) | F6 | (N) | (___) |
| 7. | F7 | (N) | (___) | F8 | (N) | (___) | F9 | (N) | (___) |
| 8. | F10 | (N) | (___) | F11 | (N) | (___) | F12 | (N) | (___) |
| 9. | F13 | (N) | (___) | F14 | (N) | (___) | F15 | (N) | (___) |
| 10. | F16 | (N) | (___) | F17 | (N) | (___) | F18 | (N) | (___) |
| 11. | F19 | (N) | (___) | F20 | (N) | (___) | F21 | (N) | (___) |
| 12. | F22 | (N) | (___) | | | | | | |
| 13. | | | | | | | | | |
| 14. | | | | | | | | | |
| 15. | (Use function key #13 for help. This will explain indicator syntax.) | | | | | | | | |
| 16. | | | | | | | | | |
| 17. | | | | | | | | | |
| 18. | | | | | | | | | |

Figure 3-8. Function/Command Key Screen

There are two ways of assigning function/command keys:

- You can accept a key and assign a response indicator to it. When a workstation operator presses the key, the indicator is turned on and returned to your program, thus prompting some action to occur within the program.
- You can simply accept a key without assigning a response indicator to it. When a workstation operator presses the key, your program is alerted to take a particular course of action.

The difference between assigning indicators and not assigning indicators depends on how you code your program and the function you're using a key for. The indicators you specify for function keys are response indicators. They're different from any of the other indicators you supply to the SFG. The other indicators (called option indicators) are passed from your program to the SFC. Response indicators, on the other hand, are passed from the SFC to your program. Therefore, response indicators are handled differently in your program. Consult Section 7 for details.

Normally, when an operator presses a function key associated with a response indicator, no data is sent to your program. Instead, your program receives either blanks, zeros, or default constant values. (A bidirectional field that has been conditionally protected or conditionally displayed returns the value it received from your program.) In addition, if an operator presses a function key that hasn't been accepted and which screen format services doesn't use, an error message is sent to the workstation. The workstation operator must then acknowledge the message in system mode.

NOTES:

1. *It is recommended that you use only function keys 2, 3, 4, and 6 through 10. Function keys 1, 5, and 11 through 22 are already defined by the system. If you do reassign a function key that the system uses, you'll lose that system function. Appendix C lists the SFC function keys.*
2. *User-assigned function keys aren't supported in an IMS environment.*
3. *Response indicators shouldn't be specified for RPG II programs. RPG II has a special set of indicators (KA-KN, KP-KW) for the function keys. F1-F14 correspond to KA-KN; F15-22 correspond to KP-KW.*
4. *When executing SFG from the UNISCOPE 400, you may not request function key specification.*

3.4.8. Nondisplay Screen

The nondisplay screen (Figure 3-9) lets you define a single field that communicates with your program but never appears on the workstation screen. This can be used when there is some value or information that you don't want an operator to have access to.

```

1.  DESCRIBE THE SINGLE NON-DISPLAYED FIELD FOR FORMAT xxxxxxxx:
2.
3.  Length of this field is (__)
4.
5.  Value of field is      (1):
6.    1 - FIRST FIELD OUTPUT BY PROGRAM (FIELD IS BIDIRECTIONAL).
7.    2 - FIELD IS INPUT-ONLY, VALUE IS:
8.  -----
9.
10. This value is always the first INPUT FIELD which is named:  (FLD00000)
11.
12.
13.
14. NOTE: If option #1 is selected above, it is also the first output field.
15.

```

Figure 3-9. Nondisplay Screen

The nondisplayed field that you define must be the first field in the input or output work area in your program. It can be an input constant associated with the format, or it may be a variable. In the latter case, this bidirectional field is communication between the output and input logic of the program. The value is not available to the workstation operator to inspect or change.

In the former case, the input-only field is a constant that enables the program to take some action based on information stored in the format rather than in the program.

The first question the nondisplay screen asks is the length of the field. The range of acceptable entries is 1 to 80. The second question the screen asks is whether the field is bidirectional or input only. If the field is input only, you enter the constant value. The SFG field name (FLD00000) appears on line 10. As with any other field name, you have the option of renaming it, using an application oriented name.

3.5. DIALOG SCREENS

After you transmit your optional screens (or, if you didn't request optional screens, then after you transmit the characteristics screen), you're ready to set up your format. This involves three passes: in pass 1, you lay your format out on a blank screen; in pass 2, you assign editing attributes to variable data fields; in pass 3, you indicate field I/O direction and initiate dialog screens for the fields that require them. Dialogs, however, are also initiated automatically for any field defined with an exclamation point in pass 2 or defined with edit characters that can be protected. Section 4 explains how to complete passes 1 through 3 and how you initiate dialogs. This subsection introduces and defines what the various dialog functions are.

Dialog screens are used to place additional requirements on how information for certain variable data fields is to be handled at run time. They serve a purpose similar to the optional screens, only they affect individual fields rather than the whole format.

Some dialogs let you set up conditional indicators that control how a field looks or behaves under certain conditions. These indicators are values you define in your program and on the appropriate dialog screen. At run time, they act as a signal to your format, telling it that a field is to be treated differently from the other fields on the screen. These indicators work the same way as the indicators used on the optional screens. Refer to 3.4.1 for a brief explanation of what the indicators do.

Table 3-2 lists and briefly describes the eight dialog screens.

Table 3-2. Dialog Screens and Their Functions

| Dialog Screen | Function |
|---------------|---|
| 1 | Always first dialog displayed. Indicates I/O direction, field name, internal field length, edit mask, and insertion characters and asks you if additional dialog screens are required |
| 2 | Indicates field response, default input value, and replenishing value for input only fields. Indicates field response and default input value for bidirectional fields. |
| 3 | Conditional display dialog. Specifies when a field will or will not appear at a workstation screen |
| 4 | Special display properties dialog. Controls the display properties of individual fields |
| 5 | Conditional retention dialog. Indicates when the contents of a field should be retained on a workstation screen |
| 6 | Conditional protection dialog. Indicates when the contents of a field should be protected |
| 7 | Field change notification. Sets the indicator value when the field is changed on input |
| 8 | Range checking. Indicates what the valid range of values is for a field |

After transmission of the I/O screen (pass 3), dialog screen 1 is always presented automatically for the first field requesting dialogs.

If more than one field requests dialogs, the SFG handles the fields one at a time. When dialog screen 1 and any subsequent dialogs are completed for the first field, dialog screen 1 is automatically presented for the next field and so on until all the fields requesting dialogs have been satisfied. The creation of your format is then complete, and the home screen is displayed for you to choose another function.

The following are some general rules to keep in mind when you're working with dialog screens:

1. You must correct all errors (indicated by blinking or message) before transmission of the screen can occur. To ignore warning messages, position the cursor to the end of the screen and press the transmit key.
2. You may terminate the session at any point via the F1 function key. No screen format is created, and the home screen appears for you to specify another function.
3. You can initiate the HELP function at any point during dialogs.
4. Positioning of the cursor before transmitting dialog screens is the same as described for the home screen (3.6).

3.5.1. Dialog Screen 1 (General Field Characteristics)

Dialog screen 1 (Figure 3-10) is always the first dialog screen presented for any field requiring dialogs. On it you can specify an alternate field name, the I/O direction, the internal field length, edit mask, and insertion characters. It is also where you request dialog screens 3 through 7.

```

1.  display of the field
2.
3.  (FLDnnnnn) is for field use of (1):  1 OUTPUT  2 INPUT  3 BOTH
4.
5.  Internal usage is (1):  1 DISPLAY  2 PACKED  3 BINARY  4 ZONED
6.  Internal length is: (nn)
7.
8.  PLEASE INDICATE WHETHER OR NOT THE FOLLOWING ARE REQUIRED:
9.  Conditional display? (1):  1 NO  2 YES
10. Special display properties? (1):  1 NO  2 YES
11. Conditional retention? (1):  1 NO  2 YES
12. Conditional protection? (1):  1 NO  2 YES
13. Field change notification? (1):  1 NO  2 YES
14. Range checking? (1):  1 NO  2 YES
15.
16. REPLACE '!' IN THE FOLLOWING LINE WITH INSERTION CHARACTERS
17. display of the field
18.
19.

```

Figure 3-10. Dialog Screen 1

Each line of the format can be analyzed as follows:

■ Line 1

This line displays the field requesting dialogs. It appears the way it was defined in pass 2 (the type attribute screen). If the field is the only one on a line, it appears by itself. However, if one or more fields requesting dialogs are located on the same line with other fields, the entire line, up to and including the field that requires dialogs, is displayed.

For example, you could design your format with three fields on a single line like this:

NAME_____ SOCIAL SECURITY_____ AMOUNT DUE_____

If dialogs are requested for both the SOCIAL SECURITY and AMOUNT DUE fields, then line 1 of dialog screen 1 for the SOCIAL SECURITY field would look like this:

```
NAME A_____ SOCIAL SECURITY: 999!99!9999
```

Line 1 of dialog screen 1 for the AMOUNT DUE field would appear as:

```
NAME A_____ SOCIAL SECURITY: 999!99!9999 AMOUNT DUE: $99.999.99
```

■ Line 3:

```
(FLDnnnnn) IS FOR FIELD USE OF (1): 1 OUTPUT 2 INPUT 3 BOTH
```

Shows a default name for the field (FLDnnnnn) and also lets you change the I/O direction for that field from the choice made on the I/O screen.

The SFG automatically gives a name to each variable data field in the format. This name is kept in a table that can be referenced for future modification (via the modify screen for the CREATE-FROM or MODIFY functions). These names are assigned as 'FLDnnnnn' sequentially, where $0 < nnnnn < 256$. You can, however, assign your own logical field name by overwriting the default name in the parentheses. The logical name should be unique and should not be a duplicate of an SFG-provided name. It must start with an alphabetic character, must not contain an embedded blank, and must not exceed eight characters. The only acceptable special characters it can contain are: #, @, \$, ?, _ (underline), and - (hyphen).

You specify the I/O direction for the field either by accepting the default value within the parentheses or by overwriting it with one of the other acceptable values. (If you indicate that the field is an INPUT or BOTH field, the next dialog screen presented is dialog screen 2.)

■ Line 5:

```
INTERNAL USAGE IS (1): 1 DISPLAY 2 PACKED 3 BINARY 4 ZONED
```

Shows the type of internal usage. The default value for this question is normally 1. You should overwrite this value if the usage of the field within your program is anything other than display. If you specify one of the other usages for an input field, the information entered by the workstation operator is automatically converted to that usage. If you specify another usage for an output field, that information is automatically converted to display form on output. Usages specified for a bidirectional (input/output) field are handled accordingly.

The default internal usage for a field defined in pass 2 with a sign (+, -, CR, DB) is ZONED. A display field can't represent a signed number (internally).

■ Line 6:

INTERNAL LENGTH IS: (nn)

Shows the default internal length calculated by the SFG. The SFG determines internal length based on the picture string defined on the type attribute screen (!'s and other edit characters used for simple and fixed insertion are not counted) and upon the internal usage (packed, binary, zoned, or display) specified.

Depending on the internal usage specified and the internal length calculated, the SFG may generate one of the three following warning messages.

SFG40 INTERNAL LENGTH IS LESS THAN DEFAULT FOR USAGE SPECIFIED

Indicates that the internal length entered for the field is less than it should be based on the internal usage specified.

SFG41 INTERNAL LENGTH IS GREATER THAN DEFAULT FOR USAGE SPECIFIED

Indicates that the internal length entered is greater than it should be based on the internal usage specified.

SFG42 INTERNAL SIZE MAY BE INVALID. CHECK USER PROGRAM

Indicates that the internal length entered may be invalid for the internal usage specified. If, for instance, a binary field is not specified as one, two, or four bytes, this message appears.

If desired, the internal length and/or the internal usage may be changed, or the warning message may be disregarded. To disregard it, place the cursor at the end of the dialog screen and transmit.

In all cases, however, the internal size and usage provided on the dialog screen must be compatible with the data structures (data division in COBOL, input and output specifications in RPG II, etc) within the program; otherwise, an error condition may result during execution of the program.

■ Lines 8 through 14:

8. PLEASE INDICATE WHETHER OR NOT THE FOLLOWING ARE REQUIRED:

| | | | |
|---------------------------------|------|------|-------|
| 9. Conditional display? | (1): | 1 NO | 2 YES |
| 10. Special display properties? | (1): | 1 NO | 2 YES |
| 11. Conditional retention? | (1): | 1 NO | 2 YES |
| 12. Conditional protection? | (1): | 1 NO | 2 YES |
| 13. Field change notification? | (1): | 1 NO | 2 YES |
| 14. Range checking? | (1): | 1 NO | 2 YES |

List the other dialog screens you can select for the field. If you enter 2 (YES) for any of the above, the associated dialog screen is displayed after you transmit dialog screen 1 (or after you complete and transmit dialog screen 2 if the field is an input field).

NOTE:

Line 10 (Special display properties) is not available for the UNISCOPE 400.

■ Lines 16 and 17

REPLACE !'S IN THE FOLLOWING LINE BY INSERTION CHARACTERS
display of the field

The contents of lines 16 and 17 vary according to the type attributes and I/O direction of the field. If exclamation marks were used to indicate that insertion characters were required, then line 16 would display the message shown in Figure 3-10 (REPLACE !'S IN THE FOLLOWING LINE BY INSERTION CHARACTERS).

For example, suppose after pass 2 you had a field defined like this:

SOCIAL SECURITY: 999!99!999

Lines 16 and 17 of dialog screen 1 for this field would then appear as follows:

REPLACE !'S IN THE FOLLOWING LINE BY INSERTION CHARACTERS
999!99!999

You would then overwrite the exclamation marks with a dash:

REPLACE !'S IN THE FOLLOWING LINE BY INSERTION CHARACTERS
999-99-999

The dashes will be protected. They aren't counted in the internal field length and can't be overwritten when the format is being used with a program at run time.

If dialogs were required for a field requiring protection for edit characters, a different message would appear on line 17. For example, if you had a field defined as:

AMOUNT DUE: \$99,999.99

Then lines 16 and 17 would appear as:

```
USE A '/' TO MARK PROTECTED EDIT CHARS IN THE FOLLOWING LINE
$99,999.99
```

The only characters you can overwrite are the comma (,) and the decimal point (.). (The currency symbol (\$) is automatically protected.) Overwriting these characters with the slash causes that character to be protected. Protected characters act as an edit mask and are not counted in the internal length of the field.

If you choose to overwrite, the field in this example would appear as \$99/999/99. When the completed format is displayed on the workstation during its use with a program, this field appears as:

```
$ _ , _ _ _ . _ _ _
```

NOTE:

If you specify protection for a comma in a numeric field, you must also specify protection for the decimal point and vice versa.

Table 3-3 summarizes how the contents of lines 16 and 17 vary according to the way a field is defined.

Table 3-3. Contents of Lines 16 and 17 of Dialog Screen 1

| If Field Requesting Dialog | Line 16 Reads | Line 17 Displays |
|--|--|-----------------------|
| Is input, output, or both Contains one or more !'s Is one of the following types: 1. Alphabetic (AA!AAA) 2. Alphanumeric (AAA!XXX) 3. Numeric (999!999) | REPLACE !'S IN THE FOLLOWING LINE BY INSERTION CHARACTERS | The field in question |
| 1. Is input or both 2. Is numeric edited: \$99,999.99 \$9.99 | USE A '/' TO MARK PROTECTED EDIT CHARS IN THE FOLLOWING LINE | The field in question |

If the field is other than those described in Table 3-3, nothing appears on lines 16 and 17.

NOTE:

*Numeric edited fields containing any acceptable combination of the floating insertion sign \$, +, or -; the character CR or DB; or the suppression/replacement character * are always output fields, and are completely protected like all other output fields. The suppression character Z is bidirectional or output and is also completely protected when used as output only. It is for this reason that you aren't asked to specify any edit characters to be protected. The same is true for alphanumeric fields containing B or O.*

Some general rules to keep in mind when completing dialog screen 1 are:

1. You provide responses for dialog screen 1 by overwriting the default values. If no overwriting is done, the defaults are automatically accepted.
2. When you use any character other than a slash to indicate protection for a numeric edited input or bidirectional field, it is ignored.
3. Characters that can be protected in a numeric edited input or bidirectional field are the comma and the decimal point. If you specify protection for one, you must do so for the other.
4. Zero suppression of a bidirectional field does not work when the punctuation of the field is protected. Zeroes appear instead of underlines.
5. You can't replace any character other than the exclamation point with an insertion character.
6. The exclamation point may be used as an insertion character.

3.5.2. Dialog Screen 2 (Input Field Response, Default, and Replenishing)

For any input field, dialog screen 2 lets you specify the:

- @ field response
- @ default input value
- @ lowercase translation option
- @ replenishing value.

The initial replenishing value cannot be specified for bidirectional fields. If you specify YES for lowercase translation, the field is translated. If you specify NO, the screen level choice is used.

Note that this choice is not a field override option. You cannot say YES at the screen level and NO at the field level to cause no translation for a particular field. NO must be specified at the screen level to change the field level choice.

Recall from 3.3.4 that the SFC can erase input from a format (from an input only field) so that a workstation operator can enter new data. When the input is erased, the area where the input appeared is replenished, or replaced, with underlines. Dialog screen 2 provides you with the option of choosing a character string other than underlines to replenish an input only field. You may also indicate how much or how little of an input only or bidirectional field must be completed by the operator. This is referred to as the field response. You can also specify a default value for an input only or bidirectional field. This value is returned to your program when an operator doesn't enter data in the input field. Figure 3-11 shows the format of a sample dialog screen 2 display for a field called the SOCIAL SECURITY field.

```

1.  SOCIAL SECURITY: 999-99-9999
2.
3.  Field response (1):  1 OPTIONAL  2 REQUIRED  3 MUST-FILL  4 DEFAULT VALUE IS:
4.  -----
5.
6.  Lowercase translation (1):  1 Yes  2 No
7.
8.
9.  Initial replenishing value is:
10. -----
11. NOTE: For DEFAULT, alpha characters will be passed to program in upper or
12.     lower case as entered. For REPLENISH, alpha characters will be displayed
13.     as entered.

```

Figure 3-11. Dialog Screen 2

The explanations for the lines in Figure 3-11 are:

■ Line 1:

Displays the field for which the dialogs are being performed. Note that any insertion characters (such as the hyphen) that you supplied in dialog screen 1 are shown.

■ Line 3:

Permits you to overwrite the default value (1) for the field response with any of the other values listed.

OPTIONAL

Means that any or no characters may be entered for the field. If no character is entered, zero will be assumed for numeric fields and left-justified with blank padding for alphabetic and alphanumeric fields.

REQUIRED

Means that at least one character must be entered for the field. The value is right-justified with zero padding for numeric fields and left-justified with blank padding for alphabetic and alphanumeric fields.

MUST-FILL

Means that the field must be completely filled in by the workstation operator.

DEFAULT VALUE IS

If no characters are entered for the input field, a value that you supply will be used for the program. The value is defined in the following line (4) and is right-justified with zero padding for numeric fields and left-justified with blank padding for alphabetic and alphanumeric fields.

■ Line 4

May only be used if you select **DEFAULT VALUE IS** for the **FIELD RESPONSE**. You overwrite the underlines with the value that you want used in the program if the workstation operator fails to enter anything for that field. As the note at the bottom of the dialog screen indicates (Figure 3-11), alphabetic characters will be passed to the program exactly as they are entered here (in uppercase or lowercase). A numeric value is converted to the appropriate internal usage (display, packed, binary, zoned) just as it would be if the workstation operator entered it. Under certain circumstances, one of the following messages may be issued for an entry on line 4.

SFG51 DEFAULT VALUE HAS AN INVALID CHARACTER, OR IS WRONG TYPE

This is displayed if you enter a value that is not compatible with the external or the internal usage specified (e.g., the default value contains an alphabetic character when the field's internal usage is specified as packed or binary) or one that has an invalid construction (e.g., more than one sign or decimal point, no valid numeric digits for a numeric field). You must correct this value to continue.

SFG50 DEFAULT VALUE IS NON-NUMERIC OR ALL UNDER-LINES (WARNING)

This is displayed if you enter a nonnumeric value for a field defined externally as numeric or numeric-edited and internally as display or zoned. This situation is acceptable but it may not be intentional. If the value is correct, the message can be ignored. To ignore it, position the cursor to the bottom of the screen and then transmit.

SFG52 DEFAULT VALUE IS TOO LARGE TO BE PROCESSED

This is displayed if the value you enter for a numeric field is:

1. more than 40 digits for DISPLAY or ZONED internal usage;
2. more than 15 digits for PACKED DECIMAL internal usage; or
3. not within the range $-2^{31} < \text{value} < (2^{31}-1)$ for BINARY internal usage.

↓

■ Line 6

Allows you to select lowercase to uppercase translation at the field level. If you specify YES, the field is translated. If you specify NO, the screen level choice is used. As indicated by the parentheses, the default value is 1 (YES).

↑

→ ■ Line 10

Pertains to input only fields and not bidirectional fields. Displays what the replenished input field looks like. (Bidirectional fields are not replenished.) You can accept the underline as the replenishing value or specify another value by overwriting these underlines. Alphabetic characters are displayed on the format exactly as they are entered here (in uppercase or lowercase).

It's important to note that if you specify a replenish character that doesn't match the field's data type (numeric, alphanumeric, alphabetic), it could result in error. If the operator completely fills in such a field (thus overwriting the replenish character), he can successfully transmit the format. Otherwise, the SFC returns the replenish value as part of the data and the illegal characters are rejected.

In addition, the SFC doesn't translate underlines (⎵) that it detects within a user-specified replenish value to blanks or zeros. Therefore, you should not specify a replenish string that *contains* an underline for numeric or alphabetic fields. If you do and an operator doesn't key over the underline with a valid character, the SFC rejects the field. Be aware, too, that if your field is alphanumeric, the SFC won't translate the underline to a blank but it will accept it as input. In this case, it could return a meaningless character to your program.

3.5.3. Dialog Screen 3 (Conditional Field Display)

Dialog screen 3 is the conditional display dialog. It appears as shown in Figure 3-12.

Conditional display lets you specify an indicator that controls when a field appears at a workstation screen. At run time, your program can decide if it needs a particular field displayed. If it does, it can turn the display indicator on. If it doesn't, it turns the indicator off. (If you use the N nnn indicator syntax, then the opposite is true: display occurs when the indicator is off and is suppressed when the indicator is on.)

1. *display of field*
- 2.
3. Specify the conditional indicator which is to CONTROL DISPLAY of field FLDnnnnn
- 4.
5. INDICATOR VALUE: (Y ___)
- 6.
- 7.
- 8.
9. NOTE: The first character of the INDICATOR VALUE should be a 'Y' if display is
10. to occur when the indicator is on; an 'N' if display is to occur when
11. indicator is off.
- 12.
13. F-KEY 13 will provide help.
- 14.
- 15.

Figure 3-12. Dialog Screen 3

When a field is not displayed, only the display constant (if one is associated with the field) appears. (For example, if you had a field like

SOCIAL SECURITY _ _ _ _ - _ _ _ - _ _ _ _

only the SOCIAL SECURITY constant appears.) When your program turns the display of an input field off, a workstation operator is unable to enter any data to it. The value returned to your program in this case is blanks, zeros, or a default value that you specified for the field via dialog screen 2. If you turn off the display of an output or bidirectional field, your program still sends a value for it out to your format, but the SFC doesn't display it on the screen. In the case of bidirectional fields, the value your program sends to the format is returned to your program as part of the input record.

If you have an output field whose value you never want the workstation operator to see, you can unconditionally prevent its display. To do so, use the N 000 syntax for the indicator. This hides the output variable, but saves you the work of rearranging your output record.

NOTES:

1. If you initially prevent the display of an input field and then on the next output operation set a retain indicator for it, an operator won't be able to enter data to that field.
2. If a field is not displayed, the associated preceding constants will be displayed.



3.5.4. Dialog Screen 4 (Special Field Display)

Dialog screen 4 is the special display properties dialog. It lets you specify:

- *Default* display properties for a field
- *Conditional* display properties for a field that only take effect when an indicator is set

Instead of only one indicator, however, dialog screen 4 lets you specify as many as four indicators, each associated with a different intensity/emphasis combination. In addition, each display combination has a different priority level.

Accessing dialog screen 4 from a UNISCOPE 400 will result in an SF34 error: TOO MANY FCC's REQUIRED PER LINE. This is because line 10 of dialog screen 1 is not applicable to this terminal. Dialog screen 4 is only accessible when a 2 (YES) is keyed in on line 10 of dialog screen 1.

Figure 3-13 shows a display of dialog screen 4.

| | | | |
|-----|--|-------------------------------|--------------------|
| 1. | <i>display of field</i> | | |
| 2. | | | |
| 3. | DISPLAY PROPERTIES FOR FIELD FLDnnnnn: | | |
| 4. | | | |
| 5. | | Intensity | Emphasis: |
| 6. | Lowest precedence: | I. Default properties (A) | (_____) |
| 7. | | II. Option indicator (_ _ _) | (_____) |
| 8. | | III. Option indicator (_ _ _) | (_____) |
| 9. | | IV. Option indicator (_ _ _) | (_____) |
| 10. | Highest precedence: | V. Option indicator (_ _ _) | (_____) |
| 11. | | | |
| 12. | | | |
| 13. | | | |
| 14. | | | |
| 15. | Intensity: | Emphasis: | |
| 16. | A. NORMAL INTENSITY | 1. UNDERLINED | 4. REVERSE VIDEO |
| 17. | B. ALTERNATE (LOW) INTENSITY | 2. COLUMN SEPARATORS | 5. BLINK FIELD |
| 18. | C. NOT DISPLAYED | 3. STRIKE THRU BARS | 6. POSITION CURSOR |
| 19. | E. SAME AS FIELDS IN ERROR | | |
| 20. | | | |

Figure 3-13. Dialog Screen 4

When dialog screen 4 first appears, line 6 shows the default display properties that have already been assigned for the field. This default can be one of two things:

1. The value assigned by the SFG
2. The value you assigned on the special display screen (3.4.4)

In Figure 3-13, the defaults shown are the SFG defaults for a bidirectional field:

```

DISPLAY PROPERTIES FOR FIELD FLDnnnnn:
                                Intensity:      Emphasis:
Lowest precedence:  I. Default properties    (A)      (_____)
  
```

The A refers to the key that appears at the bottom of the dialog screen; it signifies normal intensity. Since the emphasis field is left blank, it means there is no special emphasis for the field. You can specify a different default for the field by overwriting the SFG default with your own choice of display properties.

Every display combination you specify can have as many as six different types of emphasis and one type of intensity. To enter an intensity, refer to the key at the bottom of the screen and then enter the letter representing the intensity you want in the column marked INTENSITY. You do the same for the emphasis by entering the numbers for the different types of emphasis in the field under EMPHASIS.

On lines 7 through 10, you can specify up to four indicators and corresponding intensity/emphasis combinations. At run time, the intensity and emphasis of the field depend on which indicator is on. The indicators are assigned priority levels. This way, if more than one indicator is on at run time, the one with the highest priority level prevails. The default always has the lowest priority, followed by II OPTION IND on up to V OPTION IND, which has the highest priority level.

Preceding each indicator number you must supply a Y or N, depending on how you want the indicator to trigger a particular display. The (Y _ _ _) syntax implies that when an indicator is on, the field appears with the intensity/emphasis associated with it. The (N _ _ _) syntax implies that a field appears with a particular intensity/emphasis only when the indicator associated with it is off. (Unconditional entries are not permitted for this dialog.)

Lines 15 through 19 list the various intensities and types of emphasis you can choose from. Table 3-4 explains each one.

Table 3-4. Intensity/Emphasis Selections

| Display Property | Function/Appearance |
|--|---|
| <p>INTENSITY:</p> <p>A. NORMAL INTENSITY</p> <p>B. ALTERNATE (LOW) INTENSITY</p> <p>C. NOT DISPLAYED</p> <p>E. SAME AS FIELDS IN ERROR</p> | <p>Fields appear with normal screen intensity.</p> <p>Fields appear in low intensity, like display constants.</p> <p>Characters keyed in won't appear on the workstation screen. For example, if a field was meant for a password, an operator could enter the password without the characters actually appearing on the screen.</p> <p>A field appears with the same intensity as fields entered in error. This intensity is not permitted for output fields.</p> |
| <p>EMPHASIS:</p> <p>1. UNDERLINED</p> <p>2. COLUMN SEPARATORS</p> <p>3. STRIKE THRU BARS</p> <p>4. REVERSE VIDEO</p> <p>5. BLINK FIELD</p> <p>6. POSITION CURSOR</p> | <p>Fields are displayed with underscores.</p> <p>Fields are highlighted with the column separator character ().</p> <p>Fields appear with strike-thru bars through the characters.</p> <p>Characters in a field appear black on a green video background.</p> <p>Field is highlighted by blinking.</p> <p>You can set up your format (using an indicator) so that the cursor is automatically positioned at a particular field. Normally, the cursor is positioned at the first input or bidirectional field. If the POSITION CURSOR indicator for more than one field is set, the cursor is positioned at the first of those fields. You can't specify POSITION CURSOR for output fields.</p> |

NOTE:

Not all devices support every display attribute. See Appendix E for a list of what devices support what attributes.

3.5.5. Dialog Screen 5 (Conditional Field Retention)

This is the conditional retention dialog screen. It let's you specify an indicator that controls when the contents of a field are retained on the workstation screen. When a field is retained, only the field's display attributes (intensity and emphasis) are rewritten to the workstation screen. Figure 3-14 is a display of dialog screen 5.

```
1.  display of field
2.
3.  Specify the indicator you wish to CONTROL DISPLAY RETENTION for field FLDnnnnn:
4.
5.      INDICATOR VALUE: (Y ___)
6.
7.
8.
9.  NOTE: This indicator will override format-level display retention for this field,
10.      if specified. If this condition is met, an output command will affect the
11.      display attributes only.
12.
13.      F-KEY 13 will provide help.
14.
```

Figure 3-14. Dialog Screen 5

If your program sets a retain indicator for an output field, the field retains the value your program sent to it. A bidirectional field retains either the value your program supplied or the value an operator keyed in over your program value. An input field retains the data a workstation operator entered. The value retained in either a bidirectional or input field is the value that is ultimately returned to your program.

Since the retain operation allows the SFC to rewrite the field display attributes, you can highlight any field you retain. For example, when your program sets a field retain indicator, it can also set a special display indicator from dialog screen 4.

The retain option can also be specified at the format-level (3.4.6). However, the field-level retain indicator you specify on dialog screen 5 overrides the format-level retain indicator.

If you use the (Y _ _ _) indicator syntax, the contents of a field are retained on the workstation screen when the indicator is on. If you use the (N _ _ _) indicator syntax, the field is retained only when the indicator is off. (Use of the unconditional syntax is permitted, but not recommended.)

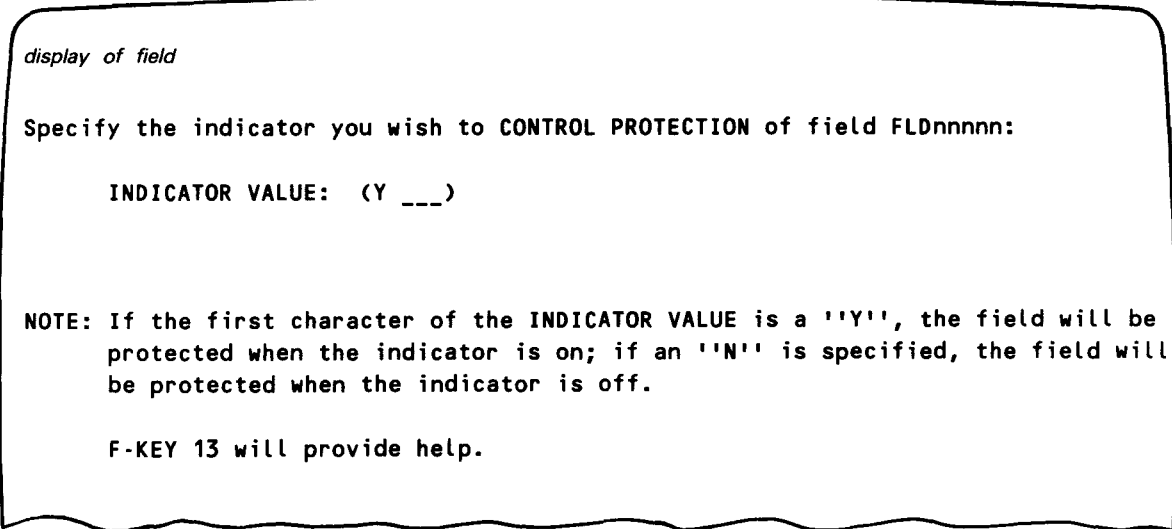
NOTE:

When a format is initially written to a screen, any retain indicators are ignored.

3.5.6. Dialog Screen 6 (Conditional Field Protection)

Dialog screen 6 is the conditional protection dialog. Conditional protection prevents an operator from entering data in an input or bidirectional field, if a protection indicator is set.

When a protection indicator is set, a workstation operator is unable to position the cursor at that field. The cursor simply skips over it. You control protection via the indicator you enter on dialog screen 6, shown in Figure 3-15.



```
1.  display of field
2.
3.  Specify the indicator you wish to CONTROL PROTECTION of field FLDnnnnn:
4.
5.      INDICATOR VALUE: (Y ___)
6.
7.
8.
9.  NOTE: If the first character of the INDICATOR VALUE is a 'Y', the field will be
10. protected when the indicator is on; if an 'N' is specified, the field will
11. be protected when the indicator is off.
12.
13.      F-KEY 13 will provide help.
14.
```

Figure 3-15. Dialog Screen 6

You can use the Y _ _ _ indicator syntax where a field is protected when the indicator is on; or you can use the N _ _ _ indicator syntax where the field is protected when the indicator is off. (Use of the Y 000 unconditional syntax is permitted, but not the unconditional N 000 syntax.)

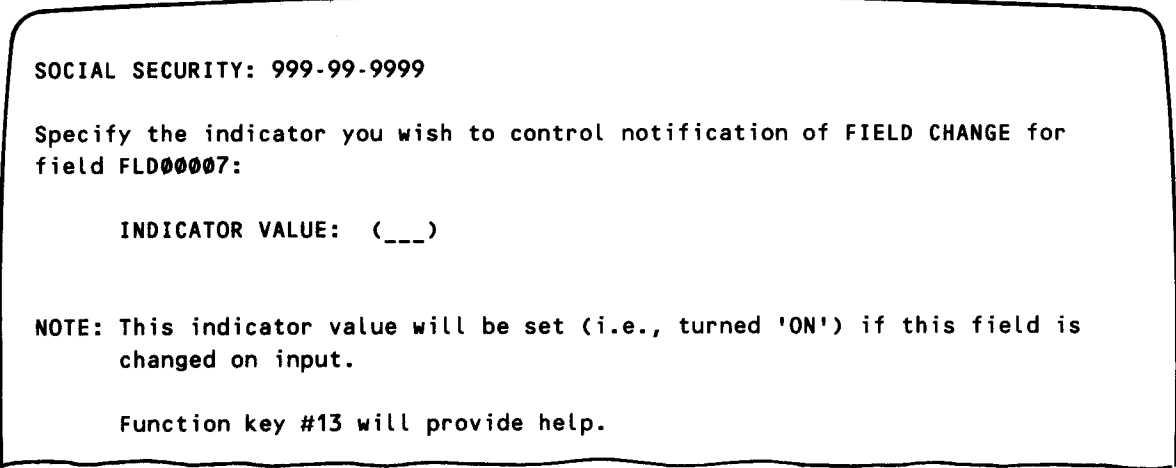
A protected bidirectional field returns either a program-supplied or user-supplied value to your program. A protected input field returns blanks, zeros, or a default constant (supplied on dialog screen 2). If your program sets both a retain and a protection indicator on, the value returned to your program is the retained value.

NOTES:

1. If you set a protection indicator on when a format is first displayed and on a subsequent output operation you reset the protection indicator, an SF08 error message is displayed.
2. On UNISCOPE 100/200 devices, a field can't be conditionally protected and retained.

3.5.7. Dialog Screen 7 (Field Change Notification)

Dialog screen 7 is the notification of field change dialog. This screen is presented when notification of field change is requested on dialog screen 1 and does not apply to output only variables. Figure 3-16 shows a dialog 7 display for a sample field called the SOCIAL SECURITY FIELD.



1. SOCIAL SECURITY: 999-99-9999
2.
3. Specify the indicator you wish to control notification of FIELD CHANGE for
4. field FLD00007:
5.
6. INDICATOR VALUE: (___)
7.
8.
9. NOTE: This indicator value will be set (i.e., turned 'ON') if this field is
10. changed on input.
11.
12. Function key #13 will provide help.

Figure 3-16. Dialog Screen 7

The indicator value on line 6 must be specified (you must enter a number ranging from 1 to 255). This indicator value will be set (i.e., turned on) if this field is changed on input.

3.5.8. Dialog Screen 8 (Input Field Range Checking)

Dialog screen 8 lets you set range check values for input and bidirectional fields. Figure 3-17 shows a dialog screen 8 display for a sample field called the SOCIAL SECURITY FIELD.

| | |
|-----|--|
| 1. | SOCIAL SECURITY: 999-99-9999 |
| 2. | |
| 3. | Specify type of CHECK and RANGE VALUE(S) for field FLD00007: |
| 4. | |
| 5. | Range check (1): 1 BETWEEN 2 LESS THAN 3 GREATER THAN 4 EQUAL |
| 6. | 5 NOT EQUAL 6 LESS/EQUAL 7 GREATER/EQUAL |
| 7. | Range value(s): |
| 8. | ---'---'--- |
| 9. | ---'---'--- |
| 10. | |
| 11. | |
| 12. | More? (1): 1 NO 2 YES |
| 13. | |

Figure 3-17. Dialog Screen 8

Range checking simply means that data an operator enters in an input or bidirectional field is checked to see if it is within range (between, greater than, less than, etc) of a set of values that you specify. You can specify range checking for numeric, alphanumeric, and alphabetic fields. The range check values themselves can be numeric or alphabetic. You select the type of range check on line 5 and supply the range value on line 8 (and line 9 if you specified BETWEEN).

You should be aware that range values displayed on dialog screen 8 will be incorrect when performing a MODIFY or CREATE-FROM operation on a packed, binary, or zoned field for which range values have been defined. This does not affect input processing; therefore, type in the desired range values and transmit.

NOTE:

For BETWEEN range checks, you must enter two values, and the first value must be less than the second value.

The field in our example is a numeric input field defined as 999-99-9999. If BETWEEN is specified, the input value must either fall between the two values or be equal to one of the two values that you supply (overwrite) on lines 8 and 9.

| | | |
|-------------------------|-------------|-------------------|
| You select | 1 BETWEEN | on line 5. |
| You supply RANGE VALUES | 195-44-8867 | on lines 8 and 9. |
| | 295-67-3563 | |

If you choose any of the other range checks, only line 8 should be overwritten with a value.

| | | |
|------------|-------------|------------|
| You select | LESS/EQUAL | on line 5. |
| You supply | 195-44-8867 | on line 8. |

You can specify, in any combination, a maximum of 10 range checks for one field by overwriting the default value on line 12 with the value 2 (YES). Upon transmission, the screen is displayed again for the same field. You may then select another range check and supply the corresponding value.

When multiple range checks are requested, the SFG automatically performs an OR operation, which logically unites the combinations for you. Suppose you select 3 (GREATER THAN) and 5 (NOT EQUAL) as the range checks for a field. The run-time value for that field is checked, ensuring that it is either greater than or not equal to the range values you specified on line 8.

An error condition is indicated at run time if the value for a field doesn't pass the range check specified. You may make provisions within your program in the event that a workstation operator is unable to correct the error. Range checking is suppressed under the following conditions:

- An operator doesn't enter data in a field. (Even though the input value returned to your program in this case is zero or blanks and thus might not meet the range check values specified, they aren't checked.)
- An operator doesn't enter data in an input field that has a default value. The default value isn't checked.
- Data that your program sends out to a bidirectional field isn't checked.

The SFC performs range checking only after it completes other lower-level validation of input. For example, if an operator enters numeric data in an alphabetic field, the SFC detects the error before it subjects the data to a range check.

NOTE:

Lowercase to uppercase translation occurs before range checking is applied. Therefore, if you enter alphabetic range values on dialog screen 8 in lowercase and you specified lower to uppercase translation on the characteristics screen, the data an operator enters always fails.

When you transmit dialog screen 8 after specifying the tenth range check or after answering 1 (NO) to the question on line 12, dialog screen 1 is automatically redisplayed for the next field requesting dialogs. If, however, no other fields request dialog, the home screen is automatically presented for you to select another function. The creation of the format is complete.

3.5.9. Default Values for Dialog Screens

If dialogs aren't requested, certain default characteristics are automatically assumed for the field. These defaults are:

- For numeric and numeric-edited fields
 1. Field ID: SFG-provided ID is accepted.
 2. Insertion characters: None
 3. Protected edit characters: None
 4. Internal usage: Display (zoned for those fields defined with a sign (+, -, CR, DB) on the type attribute screen)
 5. Internal length: SFG-computed length is accepted.
 6. Range checking: None
 7. Conditional display: No (Field always appears.)
 8. Conditional retention: No
 9. Special display properties: No
 10. Conditional protection: No
 11. Replenishing value for I or B fields: Underlines
 12. Field response: OPTIONAL (zero for numeric fields)
 13. Fields are right-justified with zero padding.
- For alphabetic and alphanumeric edited or nonedited fields
 1. Defaults 1 – 11 for numeric fields apply to alphabetic and alphanumeric fields.
 2. Field response: OPTIONAL (blanks)
 3. Fields are left-justified with blank padding.

3.6. COMPLETING SFG SCREENS – CURSOR POSITIONING AND OVERWRITING

As soon as the SFG is activated, you get a workstation display of the home screen:

```

1. Function      (1):  1 CREATE   2 CREATE-FROM  3 MODIFY   4 DELETE
2.              5 SHOW     6 LIST        7 SPOOL    8 TERMINATE
3.
4. Old format:   Format name: (_____)
5. is in library: File name:  ($Y$FMT  )
6.              Volume:   (RES   )
7.
8.
9. New format    Format name: (_____)
10. stored in library: File name: ($Y$FMT  )
11.              Volume:   (RES   )
12. File does not exist: Allocate:  (002)   cylinders
13.              Increment:  (01)    cylinders
14.
15.
16.
17.
18. *Function keys are: F1 - GO TO HOME SCREEN      F5 - BREAKPOINT SPOOL FILE
19.                   F13 - HELP                      F14 - EXIT HELP
20.                   F20 - RESTORE SCREEN
21.

```

In general, the unprotected value appearing within parentheses following any question is the default value (response) for that question. For example, on lines 5 and 10 of the home screen, the default for FILE NAME is \$Y\$FMT. You can either accept that default or overwrite it with another acceptable value.

When overwriting, you must first position the cursor over the default value. When the home screen is initially presented, the cursor appears in the upper left-hand corner of the screen. You can control its movement by using the following keys:

Cursor to HOME key

Cursor scan keys

Tab FWD or tab back keys

The results produced by using any of these keys are essentially the same. The cursor moves, but automatically stops and positions itself over the first default value. Refer to the interactive services command and facilities user guide/programmer reference for the specifics concerning control of the cursor's movement.

You can enter another acceptable value in the parentheses or keep the default value and simply move the cursor to another position. Keep in mind that the action of overwriting causes the cursor to move automatically to the next unprotected character.

NOTE:

*In this manual, sample user entries on SFG screens are shown in reverse lettering (white characters on black background), for example: **1***

Once the home screen is completed, you transmit it by pressing the XMIT key. If, for example, the function you chose on line 1 was CREATE, the characteristics screen is the next screen that appears:

| | | | |
|-----|---|-----------|---------------------------|
| 1. | SPECIFY THE GLOBAL CHARACTERISTICS FOR FORMAT ABC : | | |
| 2. | | | |
| 3. | Error retry count: | (02) | |
| 4. | Alphabet: | (ENGLISH) | |
| 5. | | | |
| 6. | Lower case translation? | (1): | 1 YES 2 NO |
| 7. | | | |
| 8. | Screen format is | (1): | 1 ORIGINAL 2 OVERLAY |
| 9. | | | |
| 10. | Screen erase/unlock option | (1): | 1 NONE 2 REPLENISH |
| 11. | | | 3 ERASE 4 UNLOCK KEYBOARD |
| 12. | | | 5 CONDITIONAL INDICATOR |
| 13. | | | |
| 14. | Transmit all-disregard cursor position? | (1): | 1 NO 2 YES |
| 15. | | | |
| 16. | Special editing characters? | (1): | 1 NO 2 YES |
| 17. | Special display control? | (1): | 1 NO 2 YES |
| 18. | | | |
| 19. | Error message field to be defined? | (1): | 1 NO 2 YES |
| 20. | Display retention on all fields? | (1): | 1 NO 2 YES |
| 21. | Function command keys to be defined? | (1): | 1 NO 2 YES |
| 22. | | | |
| 23. | Format has a non-displayed constant? | (1): | 1 NO 2 YES |
| 24. | | | |

The rules for positioning the cursor on the home screen also apply to the characteristics screen. You can transmit the characteristics screen whenever you finish all the necessary overwriting. For example, if you only want to change a value on line 10,

| | | | |
|----------------------------|------|-------------------------|-------------------|
| Screen erase/unlock option | (2): | 1 NONE | 2 REPLENISH |
| | | 3 ERASE | 4 UNLOCK KEYBOARD |
| | | 5 CONDITIONAL INDICATOR | |

you may transmit the characteristics screen immediately after you overwrite the 1 on that line. The default values for the following lines are automatically accepted. If you change all these values, the cursor must be at the bottom of the home screen before transmission. Remember, if you don't change any of the values, you simply transmit the characteristics screen after it appears.

If you make mistakes while completing any SFG screen, transmission doesn't occur. In general, the screen is redisplayed and any errors are highlighted.

NOTE:

If you overwrite any of the values on a screen, before transmitting make sure that you position the cursor past the last default value you overwrote on the screen.

These general rules for completing the home screen and characteristics screen apply to other SFG displays, such as optional and dialog screens.

If you have difficulty completing a screen at any point, you can initiate the HELP function (3.8).

For more information concerning workstation operation, refer to the interactive services commands and facilities user guide/programmer reference.

3.7. ERROR AND WARNING MESSAGES

There are certain instances when an entry on the home screen may be syntactically correct but violates an SFG logical rule. In these instances, blinking does not occur. Instead, one of the following messages is displayed on the workstation screen.

■ SFG20 OLD FORMAT _ _ _ _ _ DOES NOT EXIST.

The format specified as OLD FORMAT cannot be found. You should check to see if the spelling of the format's name is correct, or you might check the contents of the format library (\$Y\$FMT or a user-assigned library), via the librarian or the FSTATUS system command, to see if that format actually exists.

■ SFG11 FORMAT _ _ _ _ _ IS LARGER THAN CURRENT SCREEN.

This message indicates the format is larger than the screen size. For example, a format of 24 lines cannot be displayed and consequently cannot be modified on a workstation screen with a 12-line capacity.

These messages (SFG20 and SFG11) require corrective action. You must enter another format name before you can transmit the home screen. Messages SFG21 and SFG23 are only warning messages. Corrective action may be taken; however, no action is required.

■ SFG21 NEW FORMAT _____ ALREADY EXISTS. TO REPLACE, TRANSMIT.

This message indicates that the current format name is the same as an existing format name. If you don't want the old format name replaced, change the name of the new format. If you do want the old format replaced, position the cursor to the end of the format and transmit.

■ SFG22 FC TYPE ELEMENT NOT FOUND FOR FORMAT _____

This message indicates the format element type "F" exists but the "FC" type does not exist. Use the MODIFY option to create your new format using the old format name.

■ SFG23 (WARNING) _____ LANGUAGE TABLE NOT FOUND IN FRMTOUT.

This message indicates that a language other than English was specified on the home screen and that the output file into which the SFG is to write the format does not have a character set translation for that language.

Again, this is a warning message only. You may go ahead and build the format (by positioning the cursor to the bottom of the screen and transmitting); however, note that a translation table must be supplied for the file before the SFC can use the format.

■ SFG90 FUNCTION IS NOT SUPPORTED

This message indicates the function requested on the home screen, the function key, or some other specification requests a nonexistent function or feature. Correct and rerun.

The system messages programmer/operator reference lists these messages, as well as other error messages you may get while using the SFG or the SFC.

3.8. THE HELP FUNCTION

In 1.3, we mentioned that an ease-of-use feature accompanying the SFG is its capability to provide prompting for any procedure with which you might be having difficulty. This capability is the HELP function.

You may request help once the home screen is displayed and anytime during the SFG session simply by pressing the function shift key in combination with the F13 function key. You then get the appropriate screen that tells you how to complete the operation for which help was requested. You can return to the point in the SFG session where you originally left off by using the F14 function key.

Let's say that you've been presented with the home screen and selected the CREATE function, but you're not quite sure about completing the rest of the screen. If you request help, the appropriate set of instructions for completing the home screen will be displayed.

The HELP function can be initiated in this manner at any point in the SFG session. Whether you're in the middle of creating a format or modifying one, you can ask the SFG for help whenever it's necessary.



4. Creating Screen Formats

4.1. CREATING A SIMPLE SCREEN FORMAT

A simple format is one where all the SFG defaults are accepted and no optional or dialog screens are required. Such a format is included here to demonstrate the basic steps needed to create a usable screen format. Once you're familiar with this procedure, it will be easier to go on to the discussion of more complex formats later in this section.

Table 4-1 shows the sequence of steps involved in creating a simple format.

Table 4-1. Basic CREATE Function Steps (Part 1 of 2)

| Step | Result |
|--|---|
| 1. Log onto the system. | Activates the SFG and displays the home screen |
| 2. Press XMIT key. | |
| 3. Get into system mode. | |
| 4. Key in RV SFGEN. | |
| 5. Press XMIT key. | |
| 6. Accept 1 (CREATE) from home screen. | Provides name of format and where it will be stored |
| 7. Key in name of format. | |
| 8. Accept library defaults. | |
| 9. Press XMIT key. | |
| 10. Complete pass 1 – template screen. | Provides actual format layout with display constants and variable data field representation |
| 11. Press XMIT key. | |
| 12. Complete pass 2 – type attribute screen. | Provides type attributes and editing for variable data |
| 13. Press XMIT key. | |

After you transmit the home screen, the characteristics screen is automatically displayed:

| | | | |
|-----|---|------------|---------------------------|
| 1. | SPECIFY THE GLOBAL CHARACTERISTICS FOR FORMAT xxxxxxxx: | | |
| 2. | | | |
| 3. | Error retry count: | (02) | |
| 4. | Alphabet: | (ENGLISH) | |
| 5. | | | |
| 6. | Lower case translation? | (1): | 1 YES 2 NO |
| 7. | | | |
| 8. | Screen format is | (1): | 1 ORIGINAL 2 OVERLAY |
| 9. | | | |
| 10. | Screen erase/unlock option | (1): | 1 NONE 2 REPLENISH |
| 11. | | | 3 ERASE 4 UNLOCK KEYBOARD |
| 12. | | | 5 CONDITIONAL INDICATOR |
| 13. | | | |
| 14. | Transmits all-disregard cursor position? | (1): | 1 NO 2 YES |
| 15. | | | |
| 16. | Special editing characters? | (1): | 1 NO 2 YES |
| 17. | Special display control? | (1): | 1 NO 2 YES |
| 18. | | | |
| 19. | Error message field to be defined? | (1): | 1 NO 2 YES |
| 20. | Display retention on all fields? | (1): | 1 NO 2 YES |
| 21. | Function command keys to be defined? | (1): | 1 NO 2 YES |
| 22. | | | |
| 23. | Format has a non-displayed constant? | (1): | 1 NO 2 YES |
| 24. | | | |

Since this is a basic format, you accept all the defaults by simply transmitting the screen.

This gives your format the following characteristics:

- The error retry count is 2.
- English alphabet is used.
- Lowercase translation is used.
- The format is an original format, not an overlay.
- Screen erase/unlock option is NONE.
- Input will not be returned to the user program displayed on the screen.
- Your format will not use special editing characters, special display control, or an error message field.

- Your format will not retain the contents of variable data fields on the workstation screen.
- Your format will not use function keys, nor will it have a nondisplayed field.

After you transmit the characteristics screen, you're presented with a blank workstation screen to key in the format layout. This step is called pass 1.

4.1.1. Pass 1 – Providing the Layout

When the blank screen is presented, you simply lay out your format by positioning the cursor to points on the screen that you choose and by keying in any display constants and underlines representing the external length of the variable data fields. The number of lines that a format can contain, as well as the maximum length of its longest line, is limited to the physical size of the workstation screen. Also, the length of a variable data field is limited to the length of one line. Continuation of a variable data field on the following line is not permitted.

Figure 4-1 illustrates what a completed template screen might look like:

```
1.          STUDENT GRADE REPORT          __/__/82
2.  NAME:  -----
3.  STUDENT NUMBER:  -----
4.  COURSE NUMBER:  -----
5.  SECTION:  -----
6.  INSTRUCTOR:  -----
7.  FINAL COURSE GRADE:  _
```

Figure 4-1. Typical Template (Pass 1) Screen

This screen can be analyzed as follows:

- Line 1

Contains three display constants and two variable data fields. The display constants are STUDENT GRADE REPORT, the slash (/), and the /82. Both data fields are represented by two underlines (— —), indicating an external length of two character positions.

■ Line 2

Contains one display constant and one field. The display constant is NAME:, and the field is represented by 30 consecutive underlines.

■ Line 3

Contains one display constant (STUDENT NUMBER) and one variable data field (five underlines).

■ Line 4

Contains two display constants (COURSE NUMBER: and one hyphen) and two fields (two and four consecutive underlines, respectively).

■ Line 5

Contains one display constant (SECTION:) and one field (three consecutive underlines).

■ Line 6

Contains one display constant (INSTRUCTOR:) and one field (30 consecutive underlines).

■ Line 7

Contains one display constant (FINAL COURSE GRADE:) and one field (one underline).

Although completion of pass 1 is simple, there are some general notes to keep in mind.

1. The external length of a variable data field is always expressed by consecutive underlines.
2. Any character other than an underline is considered a display constant.
3. Display constants are always protected.
4. Display constants will appear on the completed format exactly the way you enter them. Thus, lowercase display constants remain in lowercase and uppercase display constants remain in uppercase. This doesn't apply to Katakana characters.

5. If you specify at least one display constant but no variable data fields, the format's creation is complete as soon as you transmit the template screen. After transmission, the home screen appears again for you to select another function.
6. If you specify at least one variable data field and press the XMIT key, that field is redisplayed for you to complete pass 2.
7. You can use the F13 function key to initiate the HELP function at any point during the completion of pass 1.
8. You can terminate pass 1 at any point by pressing the function shift in combination with the F1 key. No format is created, and the home screen is displayed for you to select another function.
9. The cursor must be at the end of the screen format you're defining before you press the XMIT key.
10. If you press XMIT before moving the cursor (before making any entries on the blank screen), the following message appears:

SFG12 NO DATA ENTERED - POSITION CURSOR TO END OF SCREEN, XMIT

If this happens, simply make the desired entries and transmit.

11. The limit for the number of variable data fields on a screen format is 255. If a greater number of variables (group of underscores) is entered, or if the format specified is too large for the generator to handle for any reason, the following message is displayed:

SFG10 NO. OF VARIABLES OR CHARACTERS IN FORMAT IS TOO LARGE.

Processing returns to the home screen.

4.1.2. Pass 2 – Specifying Type Attributes and Editing

When you transmit the finished template screen, it is automatically redisplayed for you to specify the type attributes and edit characters for all the variable data fields. The type attributes specified indicate the kind of data (numeric, alphanumeric, alphabetic) this field will receive when the format is being used to input and output data. Basically, you use COBOL-like picture clauses to define field type attributes and editing characters. Refer to Section 2 for rules governing type attributes and editing. Figure 4-2 illustrates what a completed type attribute screen looks like.

| | | | |
|----|------------------------|----------------------|----------|
| 1. | | STUDENT GRADE REPORT | 99/99/82 |
| 2. | NAME: A | ----- | |
| 3. | STUDENT NUMBER: 99999 | | |
| 4. | COURSE NUMBER: XX-9999 | | |
| 5. | SECTION: 999 | | |
| 6. | INSTRUCTOR: A | ----- | |
| 7. | FINAL COURSE GRADE: A | | |

Figure 4-2. Typical Type Attribute (Pass 2) Screen

This screen can be analyzed as follows:

- Line 1
Contains two fields that are defined as numeric: 99 and 99.
- Line 2
Contains one alphabetic field. The A in the first position indicates that the entire field is alphabetic.
- Line 3
Contains one numeric field (99999).
- Line 4
Contains one alphanumeric field (XX) and one numeric field (9999).
- Line 5
Contains one numeric field (999).
- Line 6
Contains one alphabetic field. The A in the first position indicates that the entire field is alphabetic.
- Line 7
Contains one alphabetic field (A).

Some rules that apply to the completion of the type attribute screen are:

1. You may not specify a protected area as the first part of an input field. For example, a field defined in pass 1 as _ _ _ _ may not appear as \$999 or !AAA in pass 2. Doing so will cause input to be returned incorrectly at run time.
2. The last type attribute character specified for a field is assumed for the rest of that field. For example, A_ _ _ _ denotes an entire alphabetic field.
3. An exclamation point (!) may be used in alphabetic, alphanumeric, and numeric fields to indicate that an insertion character is required. The exclamation point may not be used if any other edit characters are specified.
4. If you specified one or more special edit characters for numeric-edited fields, via the edit screen, you must use these characters on the type attribute screen. The edit screen is discussed in 3.4.3.
5. Numeric edited fields containing any acceptable combination of the floating insertion sign \$, +, or -; the character CR or DB; or the suppression/replacement character * are always output fields, and are completely protected like all other output fields. The suppression character Z is bidirectional or output, and is also completely protected when used as output only. It is for this reason that you aren't asked to specify any edit characters to be protected.
6. The preceding rule also applies to alphanumeric fields containing the edit character B or zero (0).
7. Floating insertion or suppression/replacement editing cannot be used with the CR or DB characters.
8. You can initiate the HELP function at any time during pass 2.
9. Defining an input field with more than one data type results in tab placement at the beginning of each data type. If, for example, you define an input field as AAA999 in pass 2, the cursor will stop at position 1 and then 4 when the format is displayed by a program on a workstation or UTS 400 and tabbing begins.
10. You can terminate pass 2 at any point by pressing the function shift key and the F1 function key. All entries for the type attribute screen and the preceding template screen are ignored, and no screen format is created. The home screen is automatically displayed for you to select another function.
11. If there are any errors on the type attribute screen, it will be redisplayed when you press the XMIT key, and the errors will be blinked.

12. Errors must be corrected before transmission of the screen can occur.
13. The cursor must be beyond the last character before you press the XMIT key.

After you transmit the type attribute screen, you are ready to proceed to pass 3.

4.1.3. Pass 3 – Specifying I/O Directions and Initiating Dialogs

When you transmit the type attribute screen, the template screen is displayed for you to specify the input and output directions for each field in the format. (This is also the step where you initiate dialog screens if they're required.) Figure 4-3 shows a typical initial I/O screen display.

```
                                STUDENT GRADE REPORT                                0_/0_/82
NAME: 0 _____
STUDENT NUMBER: 0 _____
COURSE NUMBER: 0 _-0 ____
SECTION: 0 __
INSTRUCTOR: 0 _____
FINAL COURSE GRADE: 0
```

Figure 4-3. Typical Initial I/O (Pass 3) Screen

The only difference between this and the template screen is that the letter O is displayed in the first position of each field.

The letter O signifies that the default I/O direction for the fields is output. You can overwrite this default with either an I for input or a B for both input and output, depending, of course, on the specific application of the format in a program. If you want to accept all of the defaults, simply transmit the I/O screen.

The effects of these I/O directions on the fields of the format are:

- O – For Output

Indicates that the field will only be used to display variable data from a program at run time. Any data displayed in an output field is always protected and cannot be altered until new output data from you program overwrites it or until the entire format is erased.

■ I – For Input

Indicates that the field will only be used for variable data entered by the workstation operator. This input will be passed to the program at run time. Input fields are unprotected (except possibly for certain edit characters).

■ B – For Both (input and output, or bidirectional, fields)

Indicates that the field will be used to display variable output or to enter variable input. Bidirectional fields are also unprotected. A bidirectional field is always expected to display output the first time the format from your program is displayed at the workstation (see 1.3.2.2).

Some general rules pertaining to the completion of the I/O screen are:

1. Numeric edited fields containing any acceptable combination of the floating sign \$, +, or -; the character CR or DB; or the suppression/replacement character * are always output fields, and are completely protected like all other output fields. The suppression character Z is bidirectional or output, and is also completely protected when used as output only. Therefore, any numeric-edited field represented in this manner in pass 2 must be defined as an output field in pass 3. Specifying anything other than an O (for output), an exclamation point (for dialogs), or an underline (for dialogs) results in an error condition (blinking).
2. The preceding rule applies to alphanumeric fields containing the characters B or O (zero).
3. You can initiate the HELP function at any time during the completion of pass 3.
4. You can terminate the I/O screen at any time by pressing the F1 function key. All entries for this and preceding passes are ignored, and no screen format is created. The home screen is automatically displayed for you to select another function.
5. If there are any errors in the I/O screen when you press the XMIT key, the screen is redisplayed, and the errors are blinked.
6. Any errors must be corrected before transmission of the screen can occur.
7. The cursor must be positioned after the last specification entered on the I/O screen before you press XMIT.
8. If dialogs are not requested, format generation is complete. The format is stored on the library specified on the home screen, and the home screen is redisplayed.

In many cases, completion of the I/O screen is the last step in creating a format. Using our simple example, we can review all three passes.

- Pass 1: You provide the format layout and transmit.

```

                                STUDENT GRADE REPORT                                __/__/82
NAME: _____
STUDENT NUMBER: _____
COURSE NUMBER:  _____
SECTION:      _____
INSTRUCTOR:  _____
FINAL COURSE GRADE:  _

```

- Pass 2: You specify type attributes and editing and then transmit.

```

                                STUDENT GRADE REPORT                                99/99/82
NAME:A _____
STUDENT NUMBER: 99999
COURSE NUMBER:  XX-9999
SECTION: 999
INSTRUCTOR: A _____
FINAL COURSE GRADE: A

```

- Pass 3: You specify input/output directions and transmit. For the example format, suppose all the fields are input fields. Pass 3 would look like this:

```

                                STUDENT GRADE REPORT                                I_/I_/82
NAME:I _____
STUDENT NUMBER: I _____
COURSE NUMBER: I _-I _____
SECTION: I ____
INSTRUCTOR: I _____
FINAL COURSE GRADE: I

```

When the I/O screen is transmitted, the format is complete and the home screen is redisplayed for you to choose another function.

4.2. CREATING MORE COMPLEX FORMATS (USING OPTIONAL AND DIALOG SCREENS)

The format created in 4.1 is a basic format. It doesn't require additional controls over any run-time characteristics, nor does it contain any edited fields. Not all formats are that simple, however. Many formats do require edited fields and are more effective when certain features of the format are altered. This subsection illustrates how formats are created with dialog and optional screens. These additional screens provide more stringent controls over a format's run-time features.

4.2.1. A Sample Format

Suppose you needed a format for a program that gathers information on purchase orders. The purchase orders are from companies that regularly place an order every month. Though the companies are the same every month, the items they order and their quantities do vary.

The user program is to display a format, outputting the name and address of the company placing an order. The order entry clerk enters the date of the order, the item name, quantity, stock number, price, and the date required. These are the format's input fields.

The program must ensure that the stock number is a valid one. If valid, the ordering information from the input fields is passed back to the program and added to a file. The input fields are then replenished to permit entry of the next order date, item, stock number, etc, for that company. If the stock number is invalid, the program blinks the number and outputs an error message to the format.

When the order entry clerk has entered all the items for a particular company, he enters an item name of all zeros (0). This directs the program to display the name and address of the next company in the file.

To meet all these specifications, the format requires these optional screens:

- The conditional erase/replenish screen. A conditional indicator supplied on this screen controls when the format's input fields are replenished. Remember that if one customer orders more than one item, only the input-only fields are replenished, not the output fields that contain the customer name and address.
- The error message screen. The format needs an error message field to notify the order entry clerk that he has entered an invalid stock number.

The dialog screens needed and the fields that need them are:

■ STOCK NUMBER

– Dialog Screen 1

Always the first dialog screen presented. Needed to prompt subsequent dialogs, specify field direction, and provide information about the field's internal usage.

– Dialog Screen 2

Always presented for input fields that require dialogs. Needed to specify operator response and any replenishing or default values.

– Dialog Screen 4

Needed to specify special display control for this field. When an order entry clerk enters an invalid stock number, the field blinks.

■ UNIT PRICE

– Dialog Screen 1

This dialog is especially important for the UNIT PRICE field, because it will be an edited field. In addition to the functions noted for the STOCK NUMBER, this dialog also lets you specify protection for a decimal point and comma associated with this field.

– Dialog Screen 2

Serves the same function as dialog screen 2 for STOCK NUMBER.

The steps needed to create the format described are summarized in Table 4-2.

Table 4-2. CREATE Function Steps Using Optional and Dialog Screens

| Step | Result |
|---|---|
| 1. Log onto the system. | Activates the SFG and displays the home screen |
| 2. Press XMIT key. | |
| 3. Get into system mode | |
| 4. Key in RV SFGEN. | |
| 5. Press XMIT key. | |
| 6. Select 1 (CREATE) from home screen. | Provides information about where the format will be stored |
| 7. Complete home screen. | |
| 8. Press XMIT key. | |
| 9. Complete characteristics screen | Provides information related to format as a whole; prompts optional screens |
| 10. Press XMIT key. | |
| 11. Optional. Complete any optional screens. | Provides more detailed information about format as a whole |
| 12. Press XMIT key. | |
| 13. Complete pass 1 – template screen. | Provides actual format layout with display constants and variable data field representation |
| 14. Press XMIT key. | |
| 15. Complete pass 2 – type attribute screen. | Provides type attributes and editing for variable data |
| 16. Press XMIT key. | |
| 17. Complete pass 3 – I/O screen. | Indicates direction of data flow between the program and the format. May initiate dialog screens for one or more fields |
| 18. Press XMIT key. | |
| 19. Optional. Complete any dialog screens. | Provides further information about specific fields |
| 20. Press XMIT key. | |
| 21. Terminate (enter 8 on line 1) the resulting home screen if no further functions are required. | Terminates the SFG session |

4.2.1.1. The Sample Home Screen and Characteristics Screen

After you activate the SFG, the home screen is displayed for you to choose a function, name your format, and indicate its library. The completed home screen for the example format is:

```

1.  FUNCTION      (1):  1 CREATE    2 CREATE-FROM  3 MODIFY    4 DELETE
2.  5 SHOW       6 LIST      7 SPOOL     8 TERMINATE
3.
4.  OLD FORMAT:   Format name: (_____)
5.  is in library: File name:  ($Y$FMT  )
6.  Volume:      (RES   )
7.
8.
9.  New Format:    Format name: (MYFORMAT)
10. stored in library: File name: (MYFILE  )
11. Volume:      (MYVOL1)
12. File does not exit: Allocate:: (005) cylinders
13. Increment:   (01)   cylinders
14.
15.
16.
17.
18. *Function keys are: F1 - GO TO HOME SCREEN  F5 - BREAKPOINT SPOOL FILE
19.                   F13 - HELP                F14 - EXIT HELP
20.                   F20 - RESTORE SCREEN
21.

```

On line 1, you accept the default (1 CREATE). The format name (MYFORMAT) is supplied on line 9. This format won't be stored on the \$Y\$FMT file. Instead, it will be stored in the file MYFILE (shown on line 10) on the volume MYVOL1 (shown on line 11). Since space for MYFILE was not previously allocated, five cylinders are specified on line 12, and the INCREMENT default of 1 is accepted.

Once you transmit the home screen, the characteristics screen appears. It is completed as shown:

| | | | |
|-----|---|------------|--|
| 1. | SPECIFY THE GLOBAL CHARACTERISTICS FOR FORMAT MYFORMAT: | | |
| 2. | | | |
| 3. | Error retry count: | (02) | |
| 4. | Alphabet: | (ENGLISH) | |
| 5. | | | |
| 6. | Lower case translation? | (1): | 1 YES 2 NO |
| 7. | | | |
| 8. | Screen format is | (1): | 1 ORIGINAL 2 OVERLAY |
| 9. | | | |
| 10. | Screen erase/unlock option | (5): | 1 NONE 2 REPLENISH 3 ERASE 4 UNLOCK KEYBOARD 5 CONDITIONAL INDICATOR |
| 11. | | | |
| 12. | | | |
| 13. | | | |
| 14. | Transmit all-disregard cursor position? | (1): | 1 NO 2 YES |
| 15. | | | |
| 16. | Special editing characters? | (1): | 1 NO 2 YES |
| 17. | Special display control? | (1): | 1 NO 2 YES |
| 18. | | | |
| 19. | Error message field to be defined? | (2): | 1 NO 2 YES |
| 20. | Display retention on all fields? | (1): | 1 NO 2 YES |
| 21. | Function command keys to be defined? | (1): | 1 NO 2 YES |
| 22. | | | |
| 23. | Format has a non-displayed constant? | (1): | 1 NO 2 YES |
| 24. | | | |

All the SFG defaults are accepted with two exceptions:

- Screen erase/unlock is by CONDITIONAL INDICATOR (line 10)
- An error message field is requested on line 19.

By specifying CONDITIONAL INDICATOR for the screen erase/unlock option (line 10), you prompt the display of the conditional erase/replenish screen. Similarly, by responding with 2 for YES on line 19, you prompt the error message screen.

After you transmit the characteristics screen, the conditional erase/replenish screen is the first of the optional screens to appear.

4.2.1.2. The Conditional Erase/Replenish Screen for MYFORMAT

Recall from 3.4.2 that you can specify a conditional indicator that controls when input fields are replenished. For the sample format, you want input fields replenished after an order entry clerk has filled them in. To this condition, suppose you assign an indicator number of 1. When you complete the conditional erase/replenish screen, it should appear as:

1. ENTER THE CONDITIONAL ACTION DESIRED AFTER INPUT FOR FORMAT MYFORMAT:
2.
3. Do you wish all input-only fields to be REPLENISHED after input based upon a
4. conditional indicator?
5.
6. (2): 1 NO 2 YES INDICATOR VALUE: (Y 1___)
7.
8. Do you wish to have the screen ERASED after input based upon a
9. conditional indicator?
10.
11. (1): 1 NO 2 YES INDICATOR VALUE: (Y ___)
12.

4.2.1.3. The Error Message Screen for MYFORMAT

After you transmit the conditional erase/replenish screen, the error message screen automatically appears. (For a complete discussion of the error message screen, see 3.4.5.) To meet the needs of the sample format, you complete the error message as shown here:

1. ENTER THE FOLLOWING INFORMATION FOR YOUR ERROR MESSAGES FOR FORMAT MYFORMAT:
2.
3. Message field name is: (ERRORMSG)
4.
5. Number of lines in error message: (1) ENTER 1 OR 2
6. Error message to be displayed on line number: (LAST) ENTER NUMBER OR 'LAST'
7.
8. Error messages conditionally displayed: indicator (Y 2__)
9.
10. DISPLAY ATTRIBUTES: Intensity (A)
11. Emphasis: (4____)
12.
13.
14.
15.
16. Intensity: Emphasis:
17. A. NORMAL INTENSITY 1. UNDERLINED 4. REVERSE VIDEO
18. B. ALTERNATE (LOW) INTENSITY 2. COLUMN SEPARATORS 5. BLINK FIELD
19. 3. STRIKE THRU BARS
20.

The name of the error message field is ERRORMSG (the default). The number of lines for the message is 1 and when it appears, it is displayed on the last line of the format. The indicator number specified is 2. Therefore, when that indicator is on, the error message is displayed. (Note that the actual contents of the error message must be defined in your program.) When the error message appears, it is displayed in reverse video.

Once you complete and transmit the error message screen, you're ready to lay the format out.

4.2.1.4. Completing Pass 1 for MYFORMAT

The rules concerning the layout (pass 1) of the sample format are the same as those discussed in 4.1.1 for the simple format. You simply key in the display constants and indicate variable data fields with underlines. The template for MYFORMAT is:

```

                                PURCHASE ORDER REPORT
CUSTOMER NAME: _____
ADDRESS: _____ STATE: __ ZIP: _____
DATE OF ORDER: __/__/__
ITEM NAME: _____
STOCK NUMBER: _____ QUANTITY: ____
UNIT PRICE: _____ DATE REQUIRED: __/__/__
  
```

4.2.1.5. Completing Pass 2 for MYFORMAT

After you transmit the template (or pass 1) screen, it is redisplayed for you to complete pass 2. In pass 2, you fill in the type attributes for the various fields. (Refer to 4.1.2 for the rules concerning pass 2.) The type attribute screen for MYFORMAT is:

```

1.                                PURCHASE ORDER REPORT
2.
3.    CUSTOMER NAME: X_____
4.    ADDRESS: X_____ STATE: A_ ZIP: 9____
5.    DATE OF ORDER: 99/99/99
6.    ITEM NAME: X_____
7.    STOCK NUMBER: 999999          QUANTITY: 999
8.    UNIT PRICE: $9,999.99        DATE REQUIRED: 99/99/99
  
```


The screen can be analyzed as follows:

■ Line 3

Contains one alphanumeric field (CUSTOMER NAME).

■ Line 4

Contains three fields: one alphanumeric field (ADDRESS), one alphabetic field (STATE), and one numeric field (ZIP).

■ Line 5

Contains three numeric fields separated by slashes (DATE OF ORDER)

■ Line 6

Contains one alphanumeric field (ITEM NAME).

■ Line 7

Contains two numeric fields (STOCK NUMBER and QUANTITY).

■ Line 8

Contains four numeric fields (one for UNIT PRICE and three, separated by slashes, for DATE REQUIRED). The UNIT PRICE field is edited (\$9,999.99). Later in the creation process (dialog screen 1) you can specify protection for the comma (,) and the decimal point (.). You may do this only for fields that are going to be used for input or both input and output. Output fields are always protected.

4.2.1.6. Completing Pass 3 for MYFORMAT

After you transmit the type attribute screen, the after image of pass 1 (the template screen) is again displayed for pass 3. In pass 3, you indicate the I/O direction for each field (see 4.1.3) and you initiate dialog screens for the fields that require them.

When the template screen first appears, the letter O (for output) is displayed in the first position of each field. Either you overwrite the O with the direction you want, or you accept it.

If you require dialogs, initiate them for a particular field by:

- Overwriting the O with an exclamation point (!)
- Overwriting the O with an underline

For certain fields, the SFG initiates dialogs for you automatically. These are:

- Fields containing the edit character !
- Fields containing the edit characters , or . that can be protected.

The pass 3 screen for MYFORMAT is completed as follows:

```
                PURCHASE ORDER REPORT
CUSTOMER NAME: O _____
ADDRESS: O _____ STATE: O_ ZIP: O ____
DATE OF ORDER: I_/I_/I_
ITEM NAME: I _____
STOCK NUMBER: ! _____ QUANTITY: I __
UNIT PRICE: I _____ DATE REQUIRED: I_/I_/I_
```

The fields CUSTOMER NAME, ADDRESS, STATE, and ZIP accept the O default because they are all output fields. The DATE OF ORDER, ITEM NAME, QUANTITY, and DATE REQUIRED fields are all defined as input fields and do not require dialogs. An exclamation point (!) in the first character position of the STOCK NUMBER field indicates that dialog screens are required for this field. Although the UNIT PRICE field is defined as an input field on the screen, dialog screens are automatically initiated for this field because it was defined as an edited field in pass 2.

Once you transmit the I/O screen, the dialog screen 1 for the first field requiring dialogs is presented. In the case of MYFORMAT, this field is the STOCK NUMBER field.

4.2.1.7. Completing Dialog Screens for MYFORMAT

Recall from 3.5.1 that on dialog screen 1 you can specify:

- An alternate field name
- The I/O direction of the field
- The internal field length
- An edit mask (for edited fields)
- Insertion characters
- Dialog screens 3 through 7

For the STOCK NUMBER field, dialog screen 1 is completed as follows:

```
1. STOCK NUMBER: 999999
2.
3. (FLD00009) is for field use of (2):  1 OUTPUT  2 INPUT  3 BOTH
4.
5. Internal usage is (1):  1 DISPLAY  2 PACKED  3 BINARY  4 ZONED
6. Internal length is: (06)
7.
8. PLEASE INDICATE WHETHER OR NOT THE FOLLOWING ARE REQUIRED:
9. Conditional display? (1):  1 NO  2 YES
10. Special display properties? (2):  1 NO  2 YES
11. Conditional retention? (1):  1 NO  2 YES
12. Conditional protection? (1):  1 NO  2 YES
13. Field change notification? (1):  1 NO  2 YES
14. Range checking? (1):  1 NO  2 YES
15.
17.
18.
19.
20.
```

The SFG default for field name is accepted, and the I/O direction is specified as input. You accept the values displayed for internal usage and internal length. Since the specifications for MYFORMAT call for blinking the STOCK NUMBER field when an entry is invalid, SPECIAL DISPLAY PROPERTIES is selected. The STOCK NUMBER field is not an edited field, nor does it require insertion characters. Therefore, lines 16 and 17 are blank.

Since the STOCK NUMBER field is defined as an input field, dialog screen 2 automatically appears after you transmit dialog screen 1. (Refer to 3.5.2 for a complete explanation of dialog screen 2's function.)

1. STOCK NUMBER: 999999
2.
3. Field response (1): 1 OPTIONAL 2 REQUIRED 3 MUST-FILL 4 DEFAULT VALUE IS:
4. -----
5.
6. Lowercase translation (1): 1 Yes 2 No
7.
8.
9. Initial replenishing value is:
10. -----
11. NOTE: For DEFAULT, alpha characters will be passed to program in upper or
12. lower case as entered. For REPLENISH, alpha characters will be displayed
13. as entered.

For the purposes of MYFORMAT, FIELD RESPONSE must be optional. (Remember that when all the items for a particular customer have been entered, all O's are entered in the ITEM NAME field. This signals the program to display the next customer name and address. Therefore, the field can't be a MUST-FILL field or a REQUIRED field, nor does it need a default value.) No special replenishing value is needed for this field; underlines are perfectly suitable for the application. Since all the dialog screen 2 defaults meet the field's requirements, simply transmit the screen as is.

The final dialog screen that appears for the STOCK NUMBER field is dialog screen 4, the special display properties dialog. (Refer to 3.5.4 for a discussion of its function.)

It should be completed as follows:

| | | | |
|-----|--|-----------------------------|--------------------|
| 1. | STOCK NUMBER: 999999 | | |
| 2. | | | |
| 3. | DISPLAY PROPERTIES FOR FIELD FLD00009: | | |
| 4. | | | |
| 5. | | Intensity: | Emphasis: |
| 6. | Lowest precedence: | I. Default properties (A) | () |
| 7. | | II. Option indicator (Y_2_) | (E) () |
| 8. | | III. Option indicator (_) | () () |
| 9. | | IV. Option indicator (_) | () () |
| 10. | Highest precedence: | V. Option indicator (_) | () () |
| 11. | | | |
| 12. | | | |
| 13. | | | |
| 14. | | | |
| 15. | Intensity: | Emphasis: | |
| 16. | A. NORMAL INTENSITY | 1. UNDERLINED | 4. REVERSE VIDEO |
| 17. | B. ALTERNATE (LOW) INTENSITY | 2. COLUMN SEPARATORS | 5. BLINK FIELD |
| 18. | C. NOT DISPLAYED | 3. STRIKE THRU BARS | 6. POSITION CURSOR |
| 19. | E. SAME AS FIELDS IN ERROR | | |
| 20. | | | |
| 21. | | | |

On line 7, you supply the indicator number that controls when the STOCK NUMBER field blinks. This number is the same as the indicator number specified for the error message field. This means that when indicator number 2 is on, it triggers two things: the display of an error message on the error message field and blinking of the invalid entry for the STOCK NUMBER. Blinking occurs when the indicator is on because the intensity chosen for this condition is SAME AS FIELDS IN ERROR (the SFG's method of highlighting a field entered in error).

Once you transmit dialog screen 4 for the STOCK NUMBER field, dialog screen 1 automatically appears for the next field requiring dialogs. This field is the UNIT PRICE field. Dialog screen 1 is automatically initiated for this field because it is an edited field and requires an edit mask for the decimal point and comma within the field. UNIT PRICE also prompts dialog screen 2 because it is an input field. The sequence of dialogs follows:

■ Dialog Screen 1

```

1.  UNIT PRICE: $9,999.99
2.
3.  (FLD00011) is for field use of (2):  1 OUTPUT  2 INPUT  3 BOTH
4.
5.  Internal usage is          (1):  1 DISPLAY  2 PACKED  3 BINARY  4 ZONED
6.  Internal length is:      (06)
7.
8.  PLEASE INDICATE WHETHER OR NOT THE FOLLOWING ARE REQUIRED:
9.  Conditional display?     (1):  1 NO      2 YES
10. Special display properties? (1):  1 NO      2 YES
11. Conditional retention?   (1):  1 NO      2 YES
12. Conditional protection?  (1):  1 NO      2 YES
13. Field change notification? (1):  1 NO      2 YES
14. Range checking?        (1):  1 NO      2 YES
15.
16. USE A '/' TO MARK PROTECTED EDIT CHARS IN THE FOLLOWING LINE
17.  $9/999/99
18.
19.
20.

```

Note that on line 17, you overwrite the comma and decimal point with a slash, indicating that these characters are to be protected.

■ Dialog Screen 2

```

1.  UNIT PRICE: $9,999.99
2.
3.  Field response (1):  1 OPTIONAL  2 REQUIRED  3 MUST-FILL  4 DEFAULT VALUE IS:
4.  $,____.____
5.
6.  Lowercase translation (1):  1 Yes  2 No
7.
8.
9.  Initial replenishing value is:
10. $,____.____
11. NOTE: For DEFAULT, alpha characters will be passed to program in upper or
12.       lower case as entered. For REPLENISH, alpha characters will be displayed
13.       as entered.

```

Since the SFG defaults are appropriate for the UNIT PRICE field, dialog screen 2 is transmitted without any modifications.

When you transmit dialog screen 2 for the UNIT PRICE field, creation of MYFORMAT is complete. At this point, the home screen is displayed for you to select another function. When you run your program, the format the order entry clerk sees is:

```

                                PURCHASE ORDER REPORT
CUSTOMER NAME:  _____
ADDRESS:  _____ STATE:  __  ZIP:  _____
DATE OF ORDER:  __/__/__
ITEM NAME:  _____
STOCK NUMBER:  _____ QUANTITY:  ____
UNIT PRICE: $ ,____.____ DATE REQUIRED:  __/__/__
  
```

4.3. OVERLAY FORMATS

Recall from 3.3.4 that an overlay format physically overlays a portion of your original format. When your program requests the display of an overlay, the overlay acts as an addition or extension to your original format. You can use it to input information to or output information from your program. Overlays can cover the entire workstation screen or just a portion of it.

4.3.1. The Need for an Overlay Format

The use and subsequent advantage of an overlay format are best illustrated with an example. Suppose that your program uses an original format that is displayed as:

```

                                SALES ORDER                                DATE:  __/__/82
CUSTOMER NAME:  _____
ADDRESS:  _____
CITY:  _____ STATE:  __  ZIP:  _____
ITEM:  _____
STOCK NUMBER:  _____
COST: $ ,____.____
QUANTITY:  ____
SALESMAN:  _____
  
```

All the fields are input fields a salesman at a workstation fills in. Every time the salesman transmits a screen, your program checks the inventory levels to make sure there's enough of the ordered item in stock. If there is, the screen format is replenished with underlines and the salesman can enter the next order. If a stock item is unavailable, your program displays an overlay format. This overlay offers the salesman alternatives to the item originally requested.

For example, a salesman enters the following order:

```

                                SALES ORDER                                DATE: 15/05/82
CUSTOMER NAME: John Doe Associates_
ADDRESS: 123 Main Street_____
CITY: Philadelphia___ STATE: PA  ZIP: 19108
ITEM: R-Widgets__
STOCK NUMBER: 1234567890
COST:$_, 50.00
QUANTITY: 10_
SALESMAN: John Smith_____

```

When your program checks the inventory level of R-Widgets, it finds only five in stock. The quantity requested is 10. Under the circumstances, your program overlays the lower half of the screen format with an overlay format. The overlay offers the salesman a choice of other similar stock items or the option of back ordering the items that are not available. The display the salesman sees is:

```

1.                                SALES ORDER                                DATE: 15/05/82
2.  CUSTOMER NAME: John Doe Associates_
3.  ADDRESS: 123 Main Street_____
4.  CITY: Philadelphia___ STATE: PA  ZIP: 19108
5.  ITEM: R-Widgets__
6.  ► ITEM IS NOT AVAILABLE
7.  YOUR ALTERNATIVES ARE:
8.     1. A-WIDGETS           2. C-WIDGETS
9.     3. E-WIDGETS           4. BACKORDER ITEM REQUESTED
10. ENTER ALTERNATE SELECTION NUMBER_

```

Lines 6 through 10 constitute the overlay format. On the overlay, the salesman makes his choice and transmits the screen. Your program then writes the original format and it is ready for the next order.

It is important to note that only one format can be active on a workstation screen at a time. What this means is that when the overlay is displayed in the example above, a salesman can enter information only on lines 6 through 10. He can't go back and make a correction to any of the information from the original format.

Now consider another example in which two overlay formats are used.

- An input format is displayed at the workstation.

```
NAME: _____  
EMPLOYEE NO.: _____
```

- The workstation operator enters information and transmits.

```
NAME: JOHN SMITH  
EMPLOYEE NO.: 54123
```

- An overlay format with corresponding employee information is selected by the program.

```
NAME: JOHN SMITH  
EMPLOYEE NO.: 54123  
DEPT #: 103 DEPT. NAME: BILLING  
DATE OF HIRE: 72/06/10  
DO YOU WISH SALARY INFO (YES, NO): ___
```

- The operator answers YES to the question on the last line and transmits. A second overlay format displays the requested information.

```
NAME: JOHN SMITH  
EMPLOYEE NO.: 54123  
CURRENT SALARY: $20000  
PREVIOUS INCREASE: 8% DATE: 790601  
GRADE: D  
MIN: $15000 MID: $19000 MAX: $23000
```

Use of an overlay can be initiated or terminated, depending on the application of your program. The important thing to remember is that an overlay format, although it can look the same as the original, is a completely separate format. It must be created separately and as a result has its own format name as well as field ID's. Logically speaking, an overlay format would be created after an original format so that its eventual positioning on the workstation screen, as it relates to the original format, can be determined.

4.3.2. Creation of an Overlay Format

An overlay format is created in much the same way as an original format. The steps for creation are:

1. Activate the SFG via LOGON and RV SFGEN.
2. Complete home screen. On the characteristics screen, use answer 2 (OVERLAY) on line 8 for SCREEN FORMAT IS.
3. Complete any requested optional screens.
4. Complete template, type attribute, and I/O screens.
5. Complete any requested dialogs.

The only difference between the creation of an overlay and an original format occurs in pass 1 – the template screen. If you want the overlay to replace, at some point, lines 6 to 10 of an original format, you must begin the overlay on line 6 of the template screen. You must also place the start-of-entry character (SOE ►) in the first position of that line. It indicates that the format is an overlay whose point of entry on an original format will be on line 6. The position of the cursor at transmission marks the end of the overlay format.

For example, to create an overlay for lines 6 through 10 of the following sample original format:

| | | | |
|----|----------------|----------------------------|----------------|
| 1. | | SALES ORDER | DATE: __/__/82 |
| 2. | CUSTOMER NAME: | _____ | |
| 3. | ADDRESS: | _____ | |
| 4. | CITY: | _____ STATE: __ ZIP: _____ | |
| 5. | ITEM: | _____ | |
| 6. | STOCK NUMBER: | _____ | |
| 7. | COST: \$ | _,____.____ | |
| 8. | QUANTITY: | ____ | |
| 9. | SALESMAN: | _____ | |

You must complete pass 1 for the overlay format as follows:

```
6.  ▶ ITEM IS NOT AVAILABLE
7.  YOUR ALTERNATIVES ARE:
8.    1. A-WIDGETS    2. C-WIDGETS
9.    3. E-WIDGETS    4. BACKORDER ITEM REQUESTED
10. ENTER ALTERNATE SELECTION NUMBER _
```

The start-of-entry character marks the beginning of the overlay, and the cursor marks its end with respect to the original format. When the template screen is transmitted, the type attribute screen is displayed for this 2-line format. Since it is a completely independent format, the type attributes that you specify may be different from those specified for the original. The same is true of I/O directions. You may also initiate dialogs for one or both of these fields.

All overlay screens must begin with the start-of-entry character (SOE), even those beginning in position (1,1). During format generation passes 2 and 3, the first line is shifted to the left one character, and the SOE character disappears. The screen is generated as specified in pass 1 with the SOE character, and no attempt should be made to shift back the correct line.

The flexibility you are afforded when creating a new or using an existing overlay format parallels that of an original. The use of one or more overlay formats for inputting and outputting can be initiated by your program. An overlay format is incorrect only if it does not physically and logically fit in with the original format to accurately reflect the requirements of the program. You should be aware that when modifying an overlay format using Modify Screen – Option 7 (CHANGE TEMPLATE ONLY – NO INTERNAL CHANGES), column 80 of the first line of the overlay format must be blank; otherwise, the format will be generated incorrectly. (See 5.1.1.4.)

4.4. CREATING A FORMAT WITH DUPLICATE LINES

When creating a screen format, you may specify that certain lines be duplicated without actually entering a line twice on the template screen.

The variable data fields on any duplicate line are totally independent from the fields on the original line. Consequently, the characteristics for these fields, such as type attributes and I/O directions, may differ.

To specify duplicate lines during creation, you enter the number of the line to be duplicated in the first position (or first and second positions for a 2-digit number) of the line where the duplicate is to appear. You could, for example, lay out the template screen in pass 1 as:

| | | | |
|----|-------|------------------------|----------------------|
| 1. | | PERSONAL CREDIT REPORT | __/__/82 |
| 2. | NAME: | ----- | |
| 3. | ADDR: | ----- | STATE: __ ZIP: ----- |
| 4. | 2 | | |
| 5. | 3 | | |
| 6. | | | |

You can type in 02 or 2 at line 4 and 03 or 3 at line 5. Either way, the system recognizes that lines 4 and 5 are duplicates of lines 2 and 3. Upon transmission, the type attribute screen is displayed as:

| | | | |
|----|-------|------------------------|----------------------|
| 1. | | PERSONAL CREDIT REPORT | __/__/82 |
| 2. | NAME: | ----- | |
| 3. | ADDR: | ----- | STATE: __ ZIP: ----- |
| 4. | NAME: | ----- | |
| 5. | ADDR: | ----- | STATE: __ ZIP: ----- |
| 6. | | | |

Since duplicate lines are independent from the originals, the type attributes selected for the fields on these lines may be the same or different from the characters selected for the originals. You could, for example, complete pass 2 like this:

| | | | |
|----|-------|------------------------|------------------------|
| 1. | | PERSONAL CREDIT REPORT | 99/99/82 |
| 2. | NAME: | A----- | |
| 3. | ADDR: | X----- | STATE: A__ ZIP: 9----- |
| 4. | NAME: | A----- | |
| 5. | ADDR: | A----- | STATE: A__ ZIP: 9----- |
| 6. | | | |

Although the first field on line 3 is specified as alphanumeric, its duplicate on line 5 is alphabetic.

This same concept can be applied when completing pass 3. I/O directions for the duplicate fields can be the same as or different from those specified for the originals. It follows that you can initiate dialogs for a duplicate field but not its original and vice versa, or you can initiate dialogs for both. You might complete pass 3 in this way:

```

1.          PERSONAL CREDIT REPORT          0_/0_/82
2.      NAME: I _____
3.      ADDR: B _____ STATE: B__ ZIP: !_____
4.      NAME: O _____
5.      ADDR: ! _____ STATE: B__ ZIP: !_____
6.

```

The rules for specifying duplicate lines are:

1. A duplicate line number must not be entered on the first line of a format.
2. A duplicate line number entered in the first position of any line must be less than the number of the line on which it is entered.
3. A duplicate line number specified in the first position (first and second positions for a 2-digit line number) must be followed by blanks for the rest of the line.
4. A blank line may not be repeated.

If you violate any of these rules, the number is treated as part of the format (a display constant) when the template screen is transmitted. No blinking occurs as in other error conditions.

Assume that you lay out the template screen in pass 1 as:

```

1.          PERSONAL CREDIT REPORT          __/__/82
2.      NAME: _____
3.      ADDR: _____ STATE: ___ ZIP: _____
4.      2
5.      3
6.      2 PAST DUE AMOUNT:
7.      10
8.      3
9.

```

Analyzing this screen, we see:

- Lines 4 and 5

Indicate that lines 2 and 3 are duplicated.

■ Line 6

Indicates that line 2 is not duplicated because blanks don't follow the number. The SFG recognizes this line as a display constant.

■ Line 7

Incorrectly specified because the number 10 is greater than the number of the line on which it appears. The number 10 is considered a display constant.

■ Line 8

Incorrectly specified because the first position contains a space. The number 3 is a display constant.

If these errors are not corrected before you transmit this screen, the type attribute screen is displayed as:

```
1.          PERSONAL CREDIT REPORT          ___/___/82
2.  NAME:  _____
3.  ADDR:  _____ STATE:  ___ ZIP:  _____
4.  NAME:  _____
5.  ADDR:  _____ STATE:  ___ ZIP:  _____
6.  2 PAST DUE AMOUNT:
7.  10
8.  3
9.
```

To make corrections, you can do one of two things:

1. Return to the home screen (function key F1) and start the format over again.
2. Complete passes 2 and 3 and any dialog screens. When the home screen is displayed, select the MODIFY function (Section 5) to correct the format.

5. Modifying Formats

5.1. CREATE-FROM AND MODIFY FUNCTIONS

The CREATE-FROM and MODIFY functions are essentially the same. CREATE-FROM allows you to create a new format by changing an existing format. The old format is not altered and is saved along with the new format in `$$FMT` or a user-assigned library file for future use.

MODIFY also allows you to create a new format by changing an existing one; however, the old format is not saved. After modification, it becomes the new format.

Among the changes you can make via the CREATE-FROM and MODIFY functions are:

- Change any format level specification from the characteristics screen.
- Change features that were defined through optional screens.
- Change the format's layout: Add fields, delete fields, and change field lengths.
- Change field type attributes.
- Change field I/O directions.
- Change field characteristics that were defined through dialogs.
- Replace lines.
- Add or insert new lines.
- Delete lines.
- Change only the format's layout (without adding or deleting fields or changing their length).

If you understand all of the steps involved in the CREATE function, the functions discussed here should present little or no difficulty. To initiate either one, you must perform the following operations:

1. Activate the SFG using LOGON and RV SFGEN. This results in a display of the home screen:

1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.

```

Function      (1)      1 CREATE      2 CREATE-FROM  3 MODIFY      4 DELETE
                    5 SHOW        6 LIST        7 SPOOL       8 TERMINATE

Old format:      Format name: (_____)
is in library:   File name:  ($Y$FMT )
                 Volume:   (RES  )

New format:      Format name: (_____)
stored in library: File name:  ($Y$FMT )
                 Volume:   (RES  )

File does not exist: Allocate: (002) cylinders
                    Increment: (01) cylinders

*Function keys are: F1 - GO TO HOME SCREEN      F5 - BREAKPOINT SPOOL FILE
                   F13 - HELP                  F14 - EXIT HELP
                   F20 - RESTORE SCREEN

```

2. Select the appropriate function code and overwrite the default on line 1.
3. Provide the old and new format names for CREATE-FROM and old name for MODIFY.

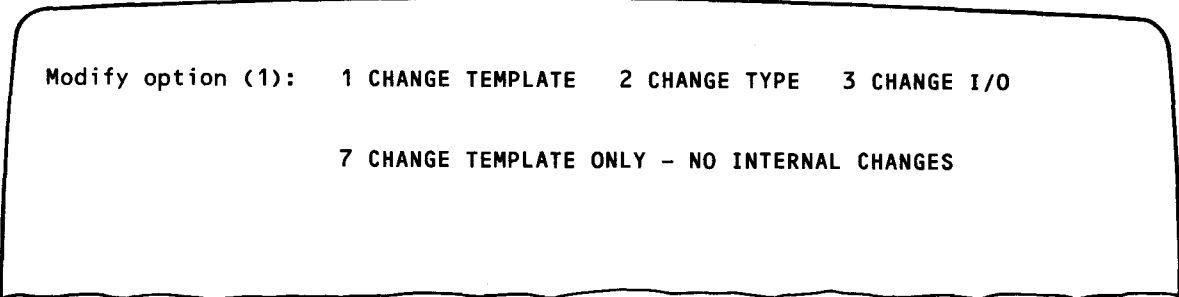
You can then make the appropriate changes to the old format.

NOTE:

When performing a modify or CREATE-FROM operation on a packed, binary, or zoned field for which range values have been defined, the range values displayed on the range values screen (dialog screen 8) will be incorrect. This does not affect input processing; therefore, type in the desired range values and transmit.

5.1.1. Modify Screen

Specific changes to an old format are initiated through the modify screen, which is displayed as in Figure 5-1.



```
1.  Modify option (1):  1 CHANGE TEMPLATE  2 CHANGE TYPE  3 CHANGE I/O
2.
3.                          7 CHANGE TEMPLATE ONLY - NO INTERNAL CHANGES
4.
5.
6.
```

Figure 5-1. Modify Screen

The modify screen (Figure 5-1) lists four options that can be used to change an existing format. You specify your selection by overwriting the default value on line 2 with any of the other acceptable values. Completing and transmitting the modify screen produces results characteristic of the option that you select.

5.1.1.1. Modify Screen – Option 1 (CHANGE TEMPLATE)

The sequence of steps for this option is quite similar to those for the CREATE function. After you transmit the modify screen, the characteristics screen automatically appears for you to make changes to your format. The values that appear on the characteristics screen are the ones you specified when you created the format. You can overwrite any of the values on the screen with new values. If you originally selected options that prompt optional screens or if you initiate new optional screens, these too will appear automatically after you transmit the characteristics screen. You can change any of the values that appear on the optional screens.

After you transmit the last optional screen (or after you transmit the characteristics screen if no optional screens were requested), the after image of the template screen for the old format is displayed without protection. You can rearrange the format or change any portion of the format layout, including display constants or the length of any field. You may also delete and add fields.

When the template screen is transmitted, the type attribute screen is displayed for you to define type attributes for every field in the format – changed as well as unchanged ones. In other words, you must complete this screen as if the format were being created for the first time whether the type attribute characters are to be the same as when the format was originally created or not.

When the type attribute screen is transmitted, the I/O screen is displayed for you to complete in the same manner by specifying I/O directions for all the fields in the format, as well as initiating dialogs for any fields that require them. If one or more dialog screens were used during creation to provide certain characteristics (such as range checking and replenishing values) for a field, you must initiate dialogs again for that field if you want these characteristics to be retained.

When all three passes and any dialog screens are completed, control returns to the home screen for you to select another function.

5.1.1.2. Modify Screen – Option 2 (CHANGE TYPE)

When you select option 2, you're first presented with the characteristics screen, where you can specify any format-level changes you wish to make. If any optional screens were requested (either when the format was created or as a result of changes to the characteristics screen), they appear after you transmit the characteristics screen. Once you've transmitted the optional screens (or the characteristics screen if no optional screens were requested), the after image of the type attribute screen for the old format is displayed.

On this display, you may only change type attribute and edit characters for any field. You may not change any display constants or delete, add, or change the length of any field. You may not rearrange the format in any way.

When this screen is transmitted, the I/O screen is displayed for you to define the I/O directions for every field in the format and to initiate dialogs as necessary. When all dialogs are completed, control returns to the home screen.

5.1.1.3. Modify Screen – Option 3 (CHANGE I/O)

If you select this option, the characteristics screen appears first, just as it does for options 1 and 2. After you transmit the characteristics screen and, if requested, any optional screens, the after image of the I/O screen for the old format is displayed. You may only change the I/O directions and initiate dialogs for the fields as necessary.

5.1.1.4. Modify Screen – Option 7 (CHANGE TEMPLATE ONLY – NO INTERNAL CHANGES)

If you select this option, the characteristics screen is the first to appear. Unlike options 1 through 3, however, there are certain changes you *cannot* make. This option only lets you alter the appearance of a format. It doesn't let you make any changes that would affect the program interface. When modifying an overlay format using this option, column 80 of the first line of the overlay format must be blank; otherwise, the format will be generated incorrectly. The changes you can make to the characteristics screen and any optional screens that appear as a result are:

- Error retry count
- Alphabet
- Edit screen: any character
- Special display screen: any parameter
- Error message screen: number of lines in error message and any parameter *except* the indicator

You may change the format's layout by doing one or both of the following:

- Change one, all, or none of the display constants and/or their location on the template screen.
- Change the location of fields on the template screen.

In either case, you may not change the *order* of the fields, nor may you add, delete, or change the length of any field. The following is the template screen for an old format.

```

                PERSONAL CREDIT REPORT
NAME: _____
ADDR: _____
ACCOUNT NUMBER: _____
PAST DUE AMOUNT: _____
NEW BALANCE: _____ DUE DATE: __/__/__
MINIMUM PAYMENT: _____

```

Modifications might result in:

```

                PERSONAL CREDIT REPORT
NAME: _____
ADDRESS: _____
ACCOUNT NUMBER: _____ PAST DUE AMOUNT: _____
NEW BALANCE: _____ DUE DATE: __/__/__ MIN PAYMENT: _____

```

Notice that two display constants have been changed: ADDR to ADDRESS and MINIMUM PAYMENT to MIN PAYMENT. The corresponding field lengths have not been altered, nor have any other field lengths. Fields have not been added or deleted, and the order of the fields is unchanged: PAST DUE AMOUNT still follows ACCOUNT NUMBER, NEW BALANCE still follows PAST DUE AMOUNT, and so on. Each field retains the same type attributes, I/O directions, and dialog-provided characteristics that were specified when the format was originally created. In other words, there are no internal field changes.

If you inadvertently change the length of a field, the following error message is displayed:

```

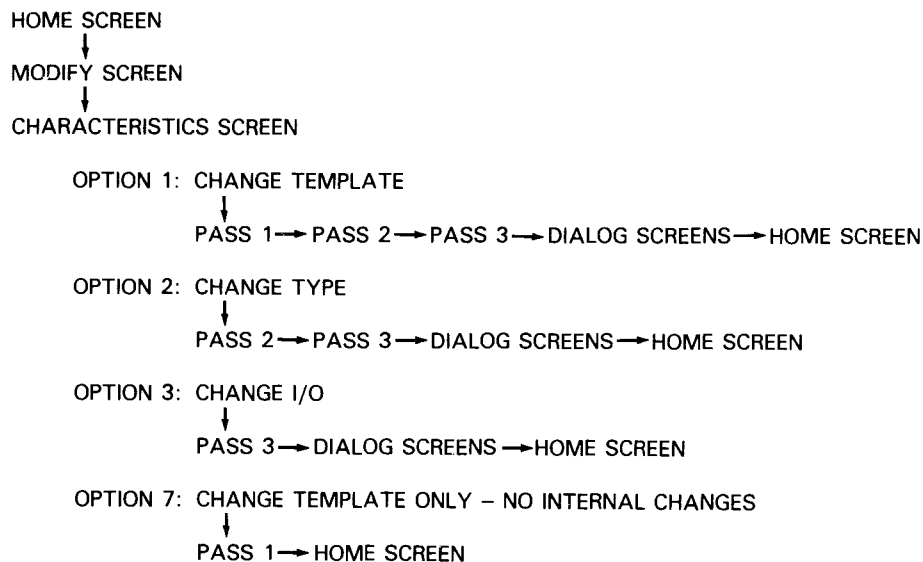
SFG30 INTERNAL FIELD HAS BEEN CHANGED.
UPDATE NOT COMPLETED.

```

After this message appears, the home screen will be displayed. Upon completion of the home and modify screens, correct the field length on the resulting template screen. Upon transmission of the template screen, whether you've rearranged it or not, modification is complete and the home screen is displayed for you to select another function.

5.2. MODIFICATION SUMMARY

The CREATE-FROM and MODIFY functions allow you to create a new format by changing an existing one. The method you use depends on the changes that you want to make and which of the previously described options affords the simplest means for making them. The following diagram provides a general summary of these options.



Keep in mind that you can terminate the CREATE-FROM or MODIFY functions (regardless of the option you're using to accomplish the changes) by using the F1 function key at any point before completion of the actual changes. You may also initiate the HELP function at any point during the CREATE-FROM or MODIFY function.

NOTES:

1. If you press XMIT before moving the cursor in pass 1 of the CREATE-FROM function, modify option 1, or modify option 7, the help screen for pass 1 will be displayed. If this happens, simply make the desired entries and press XMIT.

2. *Certain screen formats containing range check values, nondefault replenish strings, and conditional indicators may not be usable with the MODIFY/CREATE-FROM functions when beginning with modify option 2 (CHANGE TYPE) or modify option 3 (CHANGE I/O). If the Screen Format Generator detects a format that cannot be modified, this system message is displayed:*

UNABLE TO MODIFY THIS FORMAT-PLEASE USE CREATE FUNCTION

If this message appears, use modify option 1 (CHANGE TEMPLATE) to modify the existing screen format.



6. Additional Functions

6.1. DISPLAY FUNCTIONS

Each of the three display functions – SHOW, LIST, and SPOOL – gives you a display of information about an existing format.

When you want to change a format via the MODIFY function, for example, a display of the old format is useful in determining what changes you want to make. Depending on which display function you choose, you can obtain a general format display at the workstation, a listing of the detailed field characteristics at the workstation, or a printed listing of both.

The DELETE function deletes a chosen format from the library (\$Y\$FMT or a user-assigned library, see Appendix A) and the TERMINATE function ends the SFG session.

6.1.1. SHOW Function

You initiate this function by entering function code 5 on line 1 of the home screen and by entering the old format name on line 4. When the home screen is transmitted, you get a workstation display of the existing format with its field type attributes. Input-only fields will contain their replenish values to distinguish them from bidirectional fields.

You may use the TAB key to help you identify the protected fields from unprotected fields. The cursor automatically stops at the beginning of each unprotected field each time the TAB key is pressed. The low-intensity display of protected fields also helps you to make the distinction.

6.1.2. LIST Function

This function is initiated by entering function code 6 on line 1 of the home screen. Its purpose is to provide comprehensive information about a format as a whole and about the individual fields in a format. The information the LIST function provides includes:

- A summary of format-oriented information from the characteristics screen

- Details from any optional screens originally requested for the format
- The type attributes and characteristics of all fields
- Details from any dialog screens originally requested for any field
- Details about the input and output data structures showing the data
- Interface to the user program

The LIST function provides all this information by displaying a series of screens at your workstation.

When you transmit the home screen, a summary screen is displayed for you. This screen contains a synopsis of what was specified on the characteristics screen when the format was created. When you transmit the summary screen, the next screen (or series of screens) displayed depends on how the format was created.

If the format incorporated any of these options:

- Special editing characters
- Special display control
- An error message field
- Function/command keys

Then the next series of screens will include one or more of these displays:

- A special editing display
- A display control and error message display (This is one screen.)
- A function/command key display

Following transmission of these screens, you get a display of the format as it was created. If none of the above options were selected for the format, then this display of the format follows immediately after you transmit the summary screen.

After you transmit the display of the format, the next screen to appear is the list screen. This screen displays detailed characteristics of the fields in the format, such as field ID, I/O directions, and external lengths. At this time, you may also request sublist screens and conditional values screens. Sublist screens provide any field response requirements, default values, range check values, and replenishing values that were specified for any fields via dialogs. Conditional values screens display any conditional features and conditional indicators that were specified for fields, also through dialogs.

Finally, input record screens display the internal location of input fields, and output record screens display the internal location of output fields.

NOTES:

1. When you use the *LIST* function, it is displayed in two parts if your format consists of more than 12 lines.
2. If your format is using line 12 or 24, any characters in columns 79 and 80 are not displayed due to cursor positioning.
3. The preceding notes do not apply to either the *SHOW* or *SPOOL* functions nor do they affect the detailed output of the *LIST* function.

6.1.2.1. Summary Screen

Figure 6-1 shows a typical display of the first screen presented for the *LIST* function for an old format.

```

1.  NAME: FORMAT01  FORMAT SIZE: 08 X 64  DATE: 82/10/29  CREATED FROM:
2.  ALPHA:ENGLISH  FILE: $Y$FMT                                ON RES    GEN9
3.  LINE ORIGIN: 01,01  OVERLAY: NO  ERASE/UNLOCK BY: CONDITIONAL INDICATOR
4.  LOWER CASE TRANSLATION: YES  NUMBER OF VARIABLES: 0014  ERROR RETRY COUNT: 02
5.  TRANSMIT ALL, REGARDLESS OF CURSOR POSITION: NO
6.  SPECIAL EDITING CHARACTERS: YES  SPECIAL DISPLAY CONTROL: YES
7.  ERROR MSG. FIELD SPECIFIED: YES  DISPLAY RETENTION: NO
8.  FUNCTION/COMMAND KEYS DEFINED: YES
9.  FORMAT HAS A NON-DISPLAY CONSTANT: NO
10.
11.  INPUT ONLY FIELDS TO BE REPLENISHED AFTER INPUT - INDICATOR IS  Y 006
12.

```

Figure 6-1. Summary Screen

Analyzing this screen, we see:

■ Line 1

Displays the format's name (FORMAT01), its size (8 lines, the longest of which is 64 characters), its creation date (82/10/29), and the name of the format from which this format was created (if one exists).

NOTE:

The system edits the date as year/month/day.

- Line 2

Displays the language or character set (ENGLISH) that was used, the name of the screen format file (\$Y\$FMT), the volume (RES) containing that file, and your release level (GEN9, indicating release 9.0).

- Line 3

Displays the row and column number where the format begins on the screen, whether the format is an overlay format (NO), and how the format is erased and the keyboard unlocked (by CONDITIONAL INDICATOR).

- Line 4

Displays whether lowercase translation was specified (YES), the number of variable fields in the format (14), and the error retry count (02).

- Line 5

Displays whether input was specified to be returned to the user program. When YES is specified, all input will be returned to the user program displayed on the screen, regardless of where the cursor is positioned.

- Line 6

Displays whether special editing characters were specified (YES) and whether special display control was requested (YES). When special editing is YES, it prompts the appearance of the special editing display. When display control is YES, it prompts the appearance of the display control/error message display. These screens appear after you transmit the summary screen.

- Line 7

Displays whether an error message field was specified (YES) and whether display retention was specified (NO). When YES is specified for the error message field, it prompts the appearance of the display control and error message display.

- Line 8

Displays whether function/command keys were specified (YES). When YES is specified, it prompts the appearance of the function/command key display.

- Line 9

Displays whether the format has a nondisplay constant (NO). When YES, the length of the constant is also displayed on line 8, and the value of the constant is displayed on line 9.

■ Line 11

Displays the conditional indicator that controls whether input fields are replenished after input (as it was specified on the conditional erase/replenish screen).

6.1.2.2. Special Editing Display

This display is a copy of the original edit screen (3.4.3). It lists the user-defined characters, as well as the standard characters, and is shown in Figure 6-2.

| | | | |
|-----|---------------------------------|----------------------|-------------------|
| 1. | NAME: FORMAT01 | FORMAT SIZE: 08 X 64 | DATE: 80/10/13 |
| 2. | SPECIAL EDITING CHARACTERS: YES | | |
| 3. | | | |
| 4. | STANDARD CHAR : | MEANING | = CHAR TO BE USED |
| 5. | | | |
| 6. | \$: | CURRENCY SYMBOL | = (\$) |
| 7. | . | DECIMAL POINT | = (/) |
| 8. | , | THOUSANDS PUNCT. | = (,) |
| 9. | * | REPLACEMENT CHAR. | = (*) |
| 10. | CR: | CREDIT ON NEGATIVE | = (CR) |
| 11. | DB: | DEBIT ON NEGATIVE | = (DB) |
| 12. | _ : | REPLENISH CHAR. | = (_) |

Figure 6-2. Special Editing Display

6.1.2.3. Display Control and Error Message Display

This display summarizes what was originally specified on two optional screens: the special display control screen and the error message screen. It lists the display properties for display constants, output variables, input capable variables, and fields entered in error. It also lists the name of the error message field (if one was specified), along with the number of lines in the message and the line number where the message appears. A typical display control and error message display is shown in Figure 6-3.

```

1.  NAME: FORMAT01      FORMAT SIZE : 08 X 64      DATE: 82/10/13
2.  DISPLAY CONSTANTS: : (A) ( )      OUTPUT VARIABLES : (A) (1 2 3 4 5)
3.  INPUT CAPABLE VARIABLES: (A) ( )  FIELDS ENTERED IN ERROR: (B) (5 )
4.  ERROR MESSAGE TO BE DISPLAYED ON LINE NUMBER: LAST  NUMBER OF LINES: 1
5.  ERROR MSG. FIELD NAME IS: ERRORMSG  ERR.MSG DISPLAY ATTRIBUTES: (B) (5 )
6.  ERROR MESSAGE TO BE CONDITIONALLY DISPLAYED BASED ON INDICATOR (Y 001)
7.
8.          INTENSITY:                      EMPHASIS:
9.
10. A. NORMAL INTENSITY                      1. UNDERLINED                      4. REVERSE VIDEO
11. B. ALTERNATE (LOW) INTENSITY            2. COLUMN SEPARATORS              5. BLINK FIELD
12.                                         3. STRIKE THRU BARS
13.

```

Figure 6-3. Display Control and Error Message Display

6.1.2.4. Function/Command Key Display

This display is a copy of the function/command key screen. It shows key numbers and, if defined, the corresponding response for all possible function/command keys.

Figure 6-4 shows a sample display of the function/command key display.

```

1.  NAME: FORMAT01      FORMAT SIZE: 08 X 64      DATE: 80/10/13
2.  FUNCTION          COMMAND KEYS DEFINED: YES
3.
4.  KEY#  ACCEPT?  RESPONSE  KEY#  ACCEPT?  RESPONSE  KEY#  ACCEPT?  RESPONSE
5.  F1    (N)     (000)    F2    (Y)     (001)    F3    (Y)     (065)
6.  F4    (Y)     (003)    F5    (Y)     (000)    F6    (N)     (000)
7.  F7    (N)     (000)    F8    (N)     (000)    F9    (N)     (000)
8.  F10   (N)     (000)    F11   (N)     (000)    F12   (N)     (000)
9.  F13   (N)     (000)    F14   (N)     (000)    F15   (N)     (000)
10. F16   (N)     (000)    F17   (N)     (000)    F18   (Y)     (025)
11. F19   (N)     (000)    F20   (Y)     (174)    F21   (N)     (000)
12. F22   (N)     (000)

```

Figure 6-4. Function/Command Key Display

6.1.2.5. The Format Display

This display looks like the after image of pass 2, when the screen was created. It shows the type attributes for every field in the format. Figure 6-5 shows a typical format display.

```

1.          PERSONAL CREDIT REPORT          99/99/82
2.  NAME:AAAAAAAAAAAAAAAAAAAAAAAAAAAA
3.  ADDR:XXXXXXXXXXXXXXXXXXXXXXXXXXXXX  STATE:AAA ZIP:99999
4.  SOCIAL SECURITY:999!99!9999
5.  ACCOUNT NUMBER:XXX-999999
6.  AMOUNT DUE:$99.999.99
7.  BALANCE:$$,$99.99          PAYMENT DUE DATE:99/99/99

```

Figure 6-5. Screen Format

6.1.2.6. List Screen

Figure 6-6 shows the next screen to be presented for the LIST function.

```

1.          NAME: FORMAT01          FORMAT SIZE: 08 X 64          DATE: 80/10/13
2.  FIELD-ID  (Y,X)  I/O  EXTERNAL  INTERNAL  RESPN  REPLN  RANGE  COND
3.          TYP  LNGH  TYP  BYT  DIGIT
4.  FLD000001  1,55  B  N  2  D  2  2.  M  -  ( )  ( )
5.  FLD000002  1,58  B  N  2  D  2  2.  ( )  ( )
6.  NAME      2,07  I  A  30  D  30  30.  R  -  ( )
7.  ADDR      3,07  B  X  30  D  30  30.  R  -
8.  STATE     3,46  B  A  4  D  4  4.  R  -
9.  FLD000006  3,56  B  N  5  P  3  5.  O  -  ( )
10.                                     MORE? Y or N ( )
11.

```

Figure 6-6. List Screen

Line 1 of the list screen (Figure 6-6) shows the format's name (FORMAT01), its size (8 lines, the longest of which is 64 characters), and its creation date (80/10/13). The system edits the date as year/month/day. Line 2 displays the headings for the particular field characteristics listed in each column. The following is a list of the headings and corresponding column contents.

- FIELD – ID

This column contains the names of the particular format fields. These are listed in the order defined during creation. In this example, SFG-provided names, as well as user-assigned names, are listed.

- (Y, X)

This column displays the Y, X coordinates (or the physical position) of each field. The Y coordinate is the line on which the field is located, and the X coordinate indicates the position of that field on the line. The field whose ID is FLD00001, for example, is located on line 1 and starts at the 55th character position on that line.

- I/O

This column displays the I/O usage for the fields in a format, where I is for input, O is for output, and B is for both (input and/or output). Refer to 4.1.3 for a detailed discussion of I/O directions.

- EXTERNAL

This column indicates the external display type and the external display length for the format fields. The N 2 in this column means that field FLD00001 is a numeric field with an external length of 2 (underlines). The abbreviations for the external field types are:

| <u>Abbreviation</u> | <u>Meaning</u> |
|---------------------|---------------------------|
| N | Fixed numeric field |
| A | Alphabetic field |
| X | Alphanumeric field |
| NE | Numeric edited field |
| AE | Alphabetic edited field |
| XE | Alphanumeric edited field |

- INTERNAL

This column shows the internal usage of the field and the internal length. The D 2 for the first field means that its internal usage is display and its internal length is two bytes. The P 3 for the last field means that its internal usage is packed and its internal length is three bytes. Refer to 3.5.2 for a discussion of internal length and usage. The abbreviations for the internal usage types are:

| <u>Abbreviation</u> | <u>Meaning</u> |
|---------------------|----------------|
| D | Display |
| B | Binary |
| P | Packed decimal |
| Z | Zoned decimal |

■ **RESPN**

This column lists the field response requirements for input or bidirectional fields, where:

- M Means **MUST FILL** – the field must be completely filled in.
- O Means **OPTIONAL** – any or no characters may be entered.
- R Means **REQUIRED** – at least one character must be entered.
- () Means **DEFAULT** – a value has been specified.

Refer to 3.5.2 for an explanation of field response values.

■ **REPLN**

This column indicates replenishing values for input or bidirectional fields. A single character in the column shows the replenish value for the screen (either an underline or the character selected on the EDIT screen). A set of parentheses indicates that a unique replenish value has been specified for this field. Refer to 3.5.2 for an explanation of replenishing.

■ **RANGE**

This column indicates whether range checking has been requested for a field. The presence of a set of parentheses indicates that range checking has been requested. Refer to 3.5.8 for an explanation of range checking.

■ **COND**

This column indicates whether any conditional indicators were specified to control certain features of the field. A set of parentheses in this column signifies that indicators are used.

At the bottom of the screen, this continuation message is displayed:

MORE ? Y or N ()

Upon transmission of the list screen, the sublist screen (Figure 6-7) is displayed for FLD00002:

| | | | | | | | | | |
|-----|---|-------|-----|----------|----------|-------|-------|-------|------|
| 1. | FIELD-ID | (Y,X) | I/O | EXTERNAL | INTERNAL | RESPN | REPLN | RANGE | COND |
| 2. | | | | TYP LNCH | TYP BYT | DIGIT | | | |
| 3. | FLD00002 | 1,58 | B | N 2 | D 2 | 2. | (Y) | (Y) | (Y) |
| 4. | DEFAULT VALUE: | | | | | | | | |
| 5. | 99 | | | | | | | | |
| 6. | LOWER TO UPPERCASE TRANSLATION IN EFFECT. | | | | | | | | |
| 7. | RANGE: | | | | | | | | |
| 8. | BTW | 01-04 | | | | | | | |
| 9. | BTW | 06-08 | | | | | | | |
| 10. | GT | 13 | | | | | | | |

Figure 6-7. Sublist Screen

Analyzing this screen, we see:

- Lines 1 and 2

Display the same headings as lines 2 and 3 of the list screen.

- Line 3

Is a duplicate of line 4 of the list screen.

- Lines 4 and 5

Show that the default value is 99.

- Line 6

Shows that the YES option for lowercase translation is selected on dialog screen 2. For this message to appear, NO had to be selected at the screen level as well.

→ ■ Lines 7-10

Show the types of range checks and the corresponding range check values that were selected for field FLDO0002 via dialog screen 7. The abbreviations for the range check types are:

| Abbreviation | Meaning |
|--------------|-----------------------|
| BTW | Between |
| NEQ | Not equal |
| EQU | Equal |
| GT | Greater than |
| LT | Less than |
| GE | Greater than or equal |
| LE | Less than or equal |

Refer to Section 3 for an explanation of field responses, range checking, and replenishing.

↓ Upon transmission of this sublist screen, another sublist screen with the requested values is displayed for the next requesting field, and this continues until all fields are displayed. When you transmit the last sublist screen, input and output record screens are displayed.

↑ **6.1.2.8. Conditional Values Screen**

↓ This screen, if initiated, displays field features that are conditionally controlled as well as their control indicators. To display the conditional values screen, you key a Y in the set of parentheses in the COND column of the list screen. Refer to Section 6.1 for an explanation of the List Screen function.

↑ Figure 6-8 shows a typical conditional values screen display.

| | | | | | | | | |
|-----|--|------|----------|---------------------|-------|-------|-------|-------|
| 1. | FIELD-ID (Y,X) | I/O | EXTERNAL | INTERNAL | RESPN | REPLN | RANGE | COND |
| 2. | | | TYP LNCH | TYP BYT | DIGIT | | | |
| 3. | FLD00006 | 3,56 | B N 5 | P 3 5 | 0 | - | | (Y) |
| 4. | | | | | | | | |
| 5. | FIELD IS TO BE DISPLAYED IF INDICATOR 009 IS ON. | | | | | | | |
| 6. | FIELD IS TO BE RETAINED. | | | | | | | |
| 7. | FIELD IS TO BE PROTECTED IF INDICATOR 002 IS OFF. | | | | | | | |
| 8. | FIELD CHANGE NOTIFICATION VIA INDICATOR 010. | | | | | | | |
| 9. | | | | | | | | |
| 10. | FIELD DISPLAY PROPERTIES : (LISTED IN ORDER OF LOWEST TO HIGHEST PRECEDENCE) | | | | | | | |
| 11. | I. DEFAULT PROPERTIES | | | INTENSITY:NORMAL | | | | |
| 12. | | | | EMPHASIS :UNDERLINE | | | | |
| 13. | II. OPTION IND (Y 014) | | | INTENSITY:NORMAL | | | | |
| 14. | | | | EMPHASIS :R-VIDEO | | | | |
| 15. | III. OPTION IND () | | | INTENSITY: | | | | |
| 16. | | | | EMPHASIS : | | | | |
| 17. | IV. OPTION IND () | | | INTENSITY: | | | | |
| 18. | | | | EMPHASIS : | | | | |
| 19. | V. OPTION IND () | | | INTENSITY: | | | | |
| 20. | | | | EMPHASIS : | | | | |
| 21. | | | | | | | | |

Figure 6-8. Conditional Values Screen

Analyzing this screen, we see:

- Lines 1 and 2

Display the same headings as lines 2 and 3 of the list screen.

- Line 3

Is a duplicate of line 4 of the list screen.

- Line 5

Shows that the field (FLD00006) is displayed when indicator 009 is on. (This indicator would have been originally specified on dialog screen 3 when the format was created.)

- Line 6

Shows that the contents of the field are retained whenever indicator 005 is on. (This indicator would have been specified on dialog screen 5.)

- Line 7

Shows that the field is protected when indicator 002 is off. (Originally specified via dialog screen 6)

- Line 8

Shows that field change notification was specified for this field and the indicator associated with this option (as originally specified via dialog screen 7).

- Lines 10-20

Show the display properties associated with this field when indicator 014 is on (as specified via dialog screen 4).

To display the conditional values screen for the next field you requested, simply press the XMIT key. Upon transmission of the last conditional value screens, you're presented with the input and output record screens.

6.1.2.9. Input Record Screen

The input record screen displays the internal displacement in bytes for a particular field name in the input record. You may receive an input record screen upon transmission of a sublist screen or list screen if no sublist screen exists. Figure 6-9 contains an input record screen.

| | INPUT RECORD | |
|----|--------------|--------------|
| | FIELD NAME | DISPLACEMENT |
| 1. | | |
| 2. | | |
| 3. | FLD00002 | 0 |
| 4. | NAME | 2 |
| 5. | ADDRESS | 32 |
| 6. | STATE | 62 |
| 7. | FLD00006 | 66 |

Figure 6-9. Input Record Screen

Analyzing this screen, we see fields FLD00002, NAME, ADDRESS, STATE, and FLD00006 exist in the input record and that FLD00002 has an internal location of 0, NAME has an internal location of 2, ADDRESS has an internal location of 32, STATE has an internal location of 62, and FLD00006 has an internal location of 66.

6.1.2.10. Output Record Screen

The output record screen displays the internal displacement in bytes for a particular field name in the output record. You receive an output record screen upon transmission of the last input record screen. Figure 6-10 contains an output record screen.

| | OUTPUT RECORD | |
|----|---------------|--------------|
| | FIELD NAME | DISPLACEMENT |
| 1. | | |
| 2. | | |
| 3. | FLD00001 | 0 |
| 4. | ADDR | 2 |
| 5. | STATE | 32 |
| 6. | FLD00006 | 36 |

Figure 6-10. Output Record Screen

Analyzing this screen, we see the fields FLD00001, ADDR, STATE, and FLD00006 exist in the output record and that FLD00001 has an internal location of 0, ADDR has an internal location of 2, STATE has an internal location of 32, and FLD00006 has an internal location of 36. When the last output record screen is transmitted, control returns to the home screen.

NOTE:

A field specified as bidirectional (both input and output) appears on both the input and output record screens.

6.1.3. SPOOL Function

This function is initiated by entering the function code 7 on line 1 of the home screen and by supplying the old format's name on line 4 of the home screen. The SPOOL function is similar to the LIST function, except your screen formats are printed rather than displayed and responses are not required. All information regarding your format is always printed. After a job terminates, all screen formats are printed. However, if you want to have any screen formats printed before termination, you can breakpoint the spool file by pressing function key 5.

A batch job is available to spool one, some, or all of the formats in a particular library. Refer to Appendix G for details. ←

6.2. DELETE AND TERMINATE FUNCTIONS

The DELETE function allows you to delete a format from its library file. It is initiated by entering the function code 4 on line 1 and the old format's name on line 4 of the home screen. Upon transmission of the home screen, the format is displayed at the workstation along with the following question. Note that the underlines will be replaced by the name of the format to be deleted.

IS_____THE SCREEN FORMAT YOU WISH TO DELETE? _ (ENTER "Y" OR "N")

If you're sure that you want to delete the displayed format, enter Y and then press the XMIT key. If you decide not to, enter N and then press the XMIT key.

NOTE:

There is no error indication for an incorrect response. If the response to this question is anything other than Y (YES), then N (NO) is automatically assumed and the format is not deleted.

As mentioned in 3.2.1, the TERMINATE function ends the SFG session.

7. Associating Formats with a Program: Using the Screen Format Coordinator (SFC)

7.1. PROGRAMMER RESPONSIBILITIES

Now that you know the many functional capabilities that the SFG gives you to create and maintain screen formats, let's see how screen formats are incorporated into a program.

As mentioned in 1.2 and 1.4, the SFC has such responsibilities as retrieving the formats that the program is going to use, checking certain data, and controlling the flow of data between the program and the format. Although it isn't necessary for you to be concerned with the specifics of the SFC's activities, there are still certain responsibilities that you have in making the actual connection between screen formats and the program. Among these are making sure that the SFC knows where to retrieve the format, providing the proper job control statements for the format's file, including the appropriate references to the formats within the program, and incorporating any indicators that your format relies on.

7.2. JOB CONTROL

To execute a program that uses screen formats, you are responsible for providing the proper job control statements. These statements are basically the same as they would be for any program that's executed at a workstation.

You must provide a device assignment set or DVC/LFD sequence for every file that the program uses, including the workstation file, and, if the format is stored on a location other than the \$Y\$FMT library (see A.1.), the format's library file. In addition to the standard job control language (JCL), you must indicate within the DVC/LFD sequence for the workstation that screen format services are to be provided. You do this via the `// USE SFS` statement.

If your program interfaces with both the Menu Processor and Screen Format Services, you need only supply the `// USE` statement specifying the MENU option. Refer to menu services concepts and facilities, UP-9317 (current version). The SFS option for the `// USE` statement is not necessary for the dual interface, provided you perform screen selection within the program.

The format for the USE statement is:

```
//[symbol] USE SFS [ , { [format-file-lfd-1]/[format-file-lfd-2] } ]
                    [ , { SYSEMT
                      { format-file-lfd
                        }
                      } ]
[ ,initial-screen ] [ , { nnn } ]
                    [ , { 1 } ]
[ ,screen-format-1=alias-1, ..., screen-format-12=alias-12 ]
```

As you can see, most of the parameters in this statement are optional, so that if your format is stored on \$Y\$FMT the following alone is sufficient to indicate the use of screen format services:

```
// USE SFS
```

With this statement included in the DVC/LFD sequence for the workstation file, your program can retrieve any format from \$Y\$FMT simply by issuing the proper calling statement (peculiar to the program's language) and the name of the format being called. The optional parameters may be used as necessary and are explained in detail in A.1.

The job control stream to execute a program using screen formats must always include a DVC/LFD sequence, which contains the USE statement, for the workstation. Also, note that when two or more workstations access formats in the same screen format file, the DVC, USE, and LFD job control statements (with a unique LFD) must be provided for each workstation. And that, when the workstation accesses only one screen format file residing on RES, no DVC/LFD sequence is required for the file.

NOTE:

If more than one screen format file resides on RES but not in \$Y\$FMT, a DVC/LFD sequence (with a unique LFD) must be provided for the file in order for the workstation to access it.

Example 1:

```

// JOB MYJOB
.
.
.
// DVC 50
// VOL D00029 } DVC/LFD sequence for the user library
// LBL FORMAT } with a unique LFD
// LFD FMT
// DVC 200
// USE SFS,FMT } DVC/LFD sequence for
// UID W1       } the workstation
// LFD WRKSTN
.
.
.
// EXEC JOB01

```

This example shows the job control statements required for a single workstation to access a single format on a user library. There are two DVC/LFD sequences in this example: one indicating the user library where the format resides (on disk volume D00029), and one indicating a workstation, where the device code number is 200 and the name that the program uses to reference it is WRKSTN. Note that the DVC/LFD sequence for the user library provides a unique LFD for the format. That LFD is specified in the USE statement to indicate which format is to be used. The UID statement identifies the specific user communicating with the system, in this case, the user at workstation 1.

Example 2:

```

// JOB PAYROL
.
.
.
// DVC 200
// USE SFS } DVC/LFD sequence for
// UID $Y$MAS } the workstation
// LFD WRKSTN
.
.
.
// EXEC PRSNL

```

This example shows the job control statements required for a single workstation to access a single format that resides in the screen format library \$Y\$FMT on RES. It contains a DVC/LFD sequence for the workstation, where the device code number is 200 and the name that the program uses to reference the workstation is WRKSTN. The format to be used is on \$Y\$FMT; therefore, no DVC/LFD sequence is given for the file.

Example 3:

```
// DVC 50  
// VOL 123456  
// LBL FORMAT  
// LFD FMT1  
// DVC RES  
// LBL $$FMT  
// LFD FMT2  
// DVC 200 (2)  
// USE SFS,FMT1/FMT2  
// UID W1,W2  
// LFD WRKSTN
```

This example indicates formats required by your program are located in two format libraries. When your program requests a screen format, the file labeled FORMAT is searched first. If the requested format is not located, the secondary format file is searched.

For more information concerning the USE SFS statement and for more specific job control examples, see A.1. For general job control information, refer to the job control user guide.

7.3. PROGRAM CONSIDERATIONS

In addition to issuing the proper job control statements, you must make certain specifications within your program to indicate that a screen format is being used. These specifications are relatively simple and specifically related to the language in which the program is written.

You have the capability to move underlines (_) to numeric I/O fields on the screen. This will enable the operator to enter data directly without clearing the field of zeroes. User programs will not receive SF11 (USER IMPERATIVE ABNORMALLY TERMINATED) errors if the original data displayed contains underlines.

7.3.1. RPG II

Considerations for an RPG II program using screen formats relate to the following specifications forms: file, input, and output.

■ File Specifications Form

On this form, you must specify the workstation terminal as a primary file or a demand file. The block length is always omitted, and the device name (columns 40-46) must be entered as WORKSTN. You must also specify other files used by the program (excluding the format library file); however, you may not designate any of these as secondary files if the workstation file is the primary file.

■ Input Format Specifications Form

RPG II initializes the workstation program by generating a blank record. This record must be anticipated on the input format specifications form.

Every variable data field with an I/O direction of input or both input and output (bidirectional) must be identified on the input format specifications form. The field names need not correspond to those that were assigned to the actual format fields during generation.

The field location entries on the RPG II form refer to the position of these fields with respect to the input data rather than to the position of the input fields on the screen. This is simply because data is either read from the format (from input or bidirectional fields) or written to the format (to output or bidirectional fields) automatically - according to the sequence in which the variable data fields were described when the format was created.

Field lengths are counted sequentially. The FROM entry for the first input or bidirectional field is always 1, and the TO entry is always the number of the last underline in the field as defined in the format. The FROM entry for the next field picks up where the TO entry for the previous field left off. The underlines are counted in relation to this number to get the TO entry.

Let's say that the first input or bidirectional field in the format has a length of 10, and the second has a length of 12. The FROM and TO entries for the first field are 1 and 10. Entries for the second field are therefore 11 and 22. Output fields are not counted at all. If an output field is located between two input or bidirectional fields, simply bypass it.

Control levels, matching fields, and look-ahead fields may not be used with workstation input.

■ Output Format Specifications Form

Every variable data field with an I/O direction of output or bidirectional must be identified on the output format specifications form. This includes any bidirectional field that was defined on the input format specifications form.

The END POSITION entries refer to the end positions within the output data and not to their positions on the screen. For example, if there are two output or bidirectional fields with lengths of 10 and 12 respectively, the end position for the first field is 10, and that of the second field is 22.

The name of the format, as it was specified on the home screen, must be entered in the CONSTANT OR EDIT WORD field (columns 45-52) and enclosed in apostrophes. The length of the format name must be entered in column 43 with the letter K preceding it (column 42).

Use of a first-page indicator is not permitted.

Figure 7-1 gives examples of screen formats and the accompanying input/output specifications that must be made for each format.

Figure 7-1 is analyzed as follows.

■ Display 1

The screen format showing the FROM and TO positions as they relate to the length of every input and bidirectional field.

■ The Input Format Specifications Form

The input specifications showing field locations (lengths) for all input and bidirectional fields. The corresponding field names do not match the SFG-assigned field names.

■ Display 2

The screen format showing the end positions for every output and bidirectional field in the format.

■ The Output Format Specifications Form

The output specifications showing the end position (length) of every output and bidirectional format field and the format name. Field names are not SFG-assigned.

NOTES:

1. *The entries that appear on the RPG II forms in Figure 7-1 pertain only to the format data fields. Likewise, the terms input, output, and bidirectional refer only to the direction in which the data will flow between the program and the format. Files for the actual data (a master file, for example, containing names, addresses, etc) must be indicated by entries on the appropriate RPG II forms. Also, I/O directions pertaining to data on these files may deviate from the I/O directions for the format. A new address, for example, may be input to the format, but at the same time it may serve as output to (update) a master.*
2. *If during the format's creation, editing for a numeric field involved use of the decimal point or a sign (+, -, etc), you must indicate the decimal places and signs for that field on the coding form. (See 2.3 for editing rules.)*

Refer to the RPG II user guide for general information on RPG II programs.

7.3.2. COBOL

The following subsections explain:

- The major COBOL coding conventions for using screen format services
- How to code for conditional indicators, based on these conventions
- How to code for function keys and multiple workstation users

7.3.2.1. Major Coding Considerations

The major considerations for a COBOL program using screen formats relate to the following divisions:

Environment division - SPECIAL NAMES paragraph

Data division - working storage section

Procedure division - ACCEPT and DISPLAY statements

The basic considerations for each of these divisions are:

■ Environment Division

The SPECIAL NAMES paragraph is used to associate screen format services with a workstation. The basic format for the actual statement is:

```
SYSFORMAT IS mnemonic-name ASSIGN TO workstation lfd-name
```

(Additional clauses associated with this format are discussed in 7.3.2.3.)

The mnemonic-name is arbitrary, but the lfd-name must match the name specified for the workstation in your job control DVC-LFD sequence as indicated by the following:

DVC-LFD SEQUENCE FOR THE WORKSTATION

```
.
.
.
// DVC 200
// USE SFS
// UID $$$MAS
// LFD WORKSTN
.
.
.
```

THE SPECIAL NAMES STATEMENT

```
SYSFORMAT IS mnemonic-name ASSIGN TO WORKSTN
```

■ Data Division

You may define an area in the working storage section that will be used to hold input data from the format supplied by the workstation operator and output data from the COBOL program to be displayed on the format.

The data structures you define need not have the same names as those that were assigned to the fields during the format's generation. Picture clauses associated with the data structures should, however, correspond to the field lengths and type attributes that were assigned via passes 1 and 2 of the SFG session. Assumed decimal places (v) and signs (s) should be specified wherever applicable. In addition, if the data's internal usage is going to be packed, for example, this should also be indicated via the appropriate USAGE clause (USAGE COMP-3).

■ Procedure Division

Input and output activity between the program and a screen format results from the issuance of DISPLAY or ACCEPT statements in the COBOL program's procedure division.

- DISPLAY Statement

Used to display data from the program as output on the screen format

- ACCEPT Statement

Used to accept any input data supplied to the format by a workstation operator

The basic formats for both statements are as follows:

```

DISPLAY {identifier-1} [ , {identifier-2} ; ... ] UPON mnemonic-name
      {literal-1} [ {literal-2} ]
      [ USING {identifier}
          {literal} ]
ACCEPT identifier-1 [ , identifier-2 , ... ] FROM mnemonic-name
      [ USING {identifier}
          {literal} ]

```

(Additional clauses associated with these statements are discussed in 7.3.2.3.)

The identifiers following the verb DISPLAY or ACCEPT are either group item names or elementary item names for the format's input/output data structure as defined in working storage. As many identifiers as necessary may be listed after the first identifier. Literals, instead of identifiers, may be used with DISPLAY.

The mnemonic name associated with UPON must be the one specified in the SPECIAL NAMES statement.

If specified, the identifier or nonnumeric literal associated with the USING clause is a 1- to 8-character format name (as defined on the home screen). USING is required in only two cases. It is needed to specify the first screen format to be displayed at the workstation if you do not specify the initial-screen parameter in the // USE SFS job control statement. It is also needed to specify a format change in a program using more than one format. In either case, once you specify USING for a particular format, you can, but need not, specify it for subsequent transactions (ACCEPTs or DISPLAYs) with the same format.

The issuance of the first DISPLAY statement for a particular format results in the following:

- The screen format (named on the USE SFS job control statement after the USING clause) is automatically displayed on the workstation screen.
- The data indicated by the DISPLAY verb is displayed in the appropriate (output or bidirectional) fields of the format.

As a general rule, a given screen format must be displayed *upon* before it can be accepted *from*. This is true both for the initial screen format display and every time there is a visible change to that format on the workstation. An exception to this rule occurs when a screen format has no variable output data (if, for example, the format is being used strictly for input). In this case, either an ACCEPT or a DISPLAY statement may be the first transaction with a screen format.

With both ACCEPT and DISPLAY, it is mandatory that the number of characters transmitted *equal* or *exceed* the total number of characters expected by the screen format (the format size). In other words, there is no provision for selectively reading from or writing to only some of the format's fields. You may not, for example, display identifier 1, identifier 2 ... and not identifiers 3, 4, and 5 if these, as elementary (02) items in working storage, describe three of the format's output fields.

- Displaying

If the total number of characters to be displayed exceeds the format size, excess characters will be truncated. If the number of characters to be displayed is smaller than the format size, a run-time error will result.

- Accepting

If the number of characters to be accepted exceeds the format size, trailing blanks are supplied (to the program) to satisfy the request of the ACCEPT statement.

If the number of characters accepted is less than the format size, a run-time error results.

Consider the screen format in Figure 7-2.

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.

```
                SALES STATUS
SALESMAN: _____
ITEM: _____
QTY SOLD: _____
COMMISSION: $____,____.____
```

Figure 7-2. Sample SCREEN01

The field characteristics as defined during creation of the format are:

| Field Name | I/O Usage | Internal Usage | Internal Length | Type |
|------------|-----------|----------------|-----------------|------------|
| SALESMAN | Input | Display | 20 | Alphabetic |
| ITEM | Both | Display | 7 | Alphabetic |
| QTYSOLD | Both | Packed | 3 | Numeric |
| FLD00004 | Output | Display | 7 | Numeric |

The format name, as defined on the home screen, is SCREEN01 and it resides on the \$Y\$FMT library. The DVC-LFD sequence for the workstation must be included in the job control stream to execute a program using this format.

```
// JOB SALES
.
.
.
// DVC 200
// USE SFS
// UID W1
// LFD WORKSTN
.
.
.
// EXEC SUMMARY
```

Figure 7-3 shows COBOL program entries associated with SCREEN01 as they relate to the environment, data, and procedure divisions.

```

A   B
8   12
-----
ENVIRONMENT DIVISION.
.
.
.
SPECIAL NAMES
  ① SYSFORMAT IS SCREENS ASSIGN TO WORKSTN.
.
.
.
DATA DIVISION.
.
.
.
WORKING-STORAGE SECTION.
01  OUTAREA
    ② { 02 ITEM          PICTURE  9(7).
      02 QUANTITY       PICTURE  9(5) USAGE COMP-3. ③
      02 COMMISSION     PICTURE  9(5)V99. ④
01  INAREA
    02 SALESMAN         PICTURE  9(20).
    ⑤ { 02 ITEM          PICTURE  A(7).
      02 QUANTITY       PICTURE  9(5) USAGE COMP-3.
PROCEDURE DIVISION.
.
.
.
        ⑥          ⑦          ⑧
    DISPLAY OUTAREA UPON SCREENS USING SCREEN01.
.
.
.
    ⑨ DISPLAY ITEM OF OUTAREA, QUANTITY OF OUTAREA COMMISSION
      UPON SCREENS.
.
.
.
    ⑩ { ACCEPT INAREA FROM SCREENS.
      .
      .
      .
      ACCEPT SALESMAN, ITEM OF INAREA, QUANTITY OF INAREA
      FROM SCREENS.

```

Figure 7-3. COBOL Program Entries for SCREEN01 (Part 1 of 2)

NOTES:

- ① SPECIAL NAMES paragraph specifying SCREENS as the mnemonic name and WORKSTN as the lfd-name assigned via JCL for the workstation.
- ② Data names can, but do not have to, match the names assigned to the format's fields during creation.
- ③ Field was specified as packed during creation (dialog screen I); therefore, USAGE is indicated.
- ④ Editing was specified, via the type attribute screen, for this numeric field during generation of the format. Only the assumed decimal place should be indicated here.
- ⑤ The format fields corresponding to these two items (ITEM and QTYSOLD) were specified with an I/O direction of B (for both input and output, or bidirectional). Both items must be specified in OUTAREA as well as INAREA. The item names may be the same, as shown here, or different.
- ⑥ DISPLAY verb may use the group item name.
- ⑦ SCREENS is the mnemonic name used in the SYSFORMAT IS statement.
- ⑧ The USING clause specifies the first format to be used by the program. (The initial-screen parameter was not specified in the // USE SFS job control statement.) USING is not necessary, but can be specified, in subsequent DISPLAY or ACCEPT statements for the same format.
- ⑨ This DISPLAY statement lists elementary item names as identifiers.
- ⑩ ACCEPT statements showing both the group item and elementary item names as identifiers.

Figure 7-3. COBOL Program Entries for SCREEN01 (Part 2 of 2)

In this example, only one format is being used with the program. As many formats as desired may be used, however, as long as the appropriate entries are made in the data division and all DISPLAY and ACCEPT statements reference the proper format name in the USING clause.

You may not, in either the ACCEPT or DISPLAY statement, selectively list identifiers. The following would cause a run-time error since the format is expecting to receive output data for the field COMMISSION as well as ITEM and QUANTITY:

```
DISPLAY ITEM,QUANTITY UPON SCREENS USING SCREEN01
```

Since the 02 data division declarations for bidirectional fields may have the same INAREA as well as OUTAREA names (ITEM and QUANTITY), you are provided with an opportunity to specify the following or similar operations in the procedure division:

| | |
|---|----|
| A | B |
| 8 | 12 |

```
DISPLAY OUTAREA UPON SCREENS USING SCREEN01.
ACCEPT INAREA FROM SCREENS USING SCREEN01.
MOVE CORRESPONDING INAREA TO OUTAREA.
DISPLAY OUTAREA UPON SCREENS USING SCREEN01.
```

Consider now the format in Figure 7-4.

PERSONAL CREDIT REPORT

NAME: _____

ADDR: _____

ACCNT NUMBER: _____

PAST DUE AMOUNT:\$ _____

PAYMENT DUE:\$ _____

NEW BALANCE:\$ _____

ACCOUNT STATUS: _____

Figure 7-4. Sample SCREEN02

The field characteristics as defined during creation are:

| Field Name | I/O Usage | Internal Usage | Internal Length | Type |
|------------|-----------|----------------|-----------------|--------------|
| NAME | Input | Display | 25 | Alphabetic |
| ADDR | Both | Display | 30 | Alphanumeric |
| FLD00003 | Input | Display | 9 | Numeric |
| PAST-AMT | Output | Display | 7 | Numeric |
| PAYMT-DUE | Input | Display | 7 | Numeric |
| NEW-BAL | Output | Display | 7 | Numeric |
| FLD00007 | Output | Display | 25 | Alphanumeric |

The format name as defined on the home screen is SCREEN02, and it resides on the \$Y\$FMT library. The DVC-LFD sequence for the workstation is:

```

// JOB CREDIT
.
.
.
// DVC 200
// USE SFS
// UID W1
// LFD WORKSTN
.
.
.
// EXEC REPORTS

```

Figure 7-5 shows COBOL program entries that relate to the environment and data divisions.

| A | B |
|---|--|
| 8 | 12 |
| ENVIRONMENT DIVISION. | |
| . | |
| . | |
| . | |
| SPECIAL NAMES. | |
| SYSFORMAT IS FORMATFL ASSIGN TO WORKSTN | |
| . | |
| . | |
| . | |
| DATA DIVISION. | |
| . | |
| . | |
| . | |
| WORKING-STORAGE SECTION. | |
| 77 | INACTIVE PIC X(25) VALUE 'ACCOUNT INACTIVE'. |
| 77 | OVERDUE PIC X(25) VALUE 'ACCOUNT OVERDUE'. |
| 01 | OUTAREA. |
| 02 | ADDR PIC X(30). |
| 02 | PAST-AMT PIC 9(5)V99. |
| 02 | NEW-BAL PIC 9(5)V99. |
| 02 | ACCNT-STATUS PIC X(25) VALUE SPACES. |
| 01 | INAREA. |
| 02 | NAME PIC (25). |
| 02 | ADDR PIC X(30). |
| 02 | ACCNT-NUM PIC 9(9). |
| 02 | PAYMT-DUE PIC 9(5)V99. |

Figure 7-5. COBOL Program Entries for SCREEN02

In this example, the utility of being able to list several identifiers in a single ACCEPT or DISPLAY statement can easily be demonstrated. If, for example, you want the literal 'ACCOUNT INACTIVE' to be displayed as output in the ACCOUNT STATUS field (FLD00007 as defined by the SFG and ACCNT-STATUS as described in working-storage) on the format, you simply issue the following statement in the procedure division:

```
DISPLAY ADDR,PAST-AMT,NEW-BAL,INACTIVE UPON FORMATFL USING SCREEN02.
```

If you don't want to display anything for ACCOUNT STATUS on the format, you simply use:

```
DISPLAY OUTAREA UPON FORMATFL USING SCREEN02.
```

You may not, however, use:

```
DISPLAY ADDR,PAST-AMT,NEW-BAL UPON FORMATFL USING SCREEN02.
```

Refer to the 1974 ANSI COBOL programmer reference for more information concerning COBOL programming in general.

7.3.2.2. Using Indicators

You can specify two types of indicators when creating a screen format:

1. Option indicators

These affect the physical characteristics of your screen format at run time. They are output from your program to the screen format coordinator.

2. Response indicators

These are the indicators associated with function keys. They are input from the screen format to your program. (Function key indicators require additional coding that option indicators do not require. See 7.3.2.3 for details.)

For indicators to work, you must define them in your program and send their values to the SFC. You do this the same way that you define and send data to output fields or accept data from input fields on a screen format (as discussed in the previous subsection). First you define indicator fields in the WORKING-STORAGE SECTION. For option indicators, you use the DISPLAY statement to pass indicator field data to the SFC. For response indicators, you use the ACCEPT statement to pass a value from the screen format to your program.

When you organize the WORKING-STORAGE SECTION of your program, follow these guidelines:

- The area where you define option indicator fields must precede the area where you define output data fields. The indicator field area can be part of a group that includes the output data fields.
- The area where you define response indicator fields must precede the area where you define input data fields. The response indicator area can also be a part of a group that includes the input data fields.

Option indicator fields must be made a part of the DISPLAY statement along with the output data fields. Similarly, response indicator fields must be made a part of the ACCEPT statement, along with the input data fields. Where indicators are used, the SFC automatically expects the leading bytes from a DISPLAY or an ACCEPT operation to be indicators.

The following example illustrates how to handle indicators in your program.

Once again, consider the format shown in Figure 7-2:

```
                SALES STATUS
SALESMAN: _ _ _ _ _
ITEM: _ _ _ _ _
QTY SOLD: _ _ _ _ _
COMMISSION: $ _ , _ _ _ _
```

This time, suppose the format was created with option indicators 10 and 20 and response indicator 30. (To keep this explanation brief and easy to follow, the functions performed by the indicators are not discussed.) One way of defining the indicators, turning them on, and identifying them to the SFC is shown in Figure 7-6.

| | |
|---|----|
| A | B |
| 8 | 12 |

ENVIRONMENT DIVISION.

.

.

.

SPECIAL NAMES.

SYSFORMAT IS SCREENS ASSIGN TO WORKSTN.

.

.

.

DATA DIVISION.

.

.

.

WORKING STORAGE SECTION.

01 OUTAREA. (1)

02 INDICATOR-REGION

03 IND-10

PICTURE 9.

03 IND-20

PICTURE 9.

02 DATA-REGION

03 ITEM

PICTURE 9(7).

03 QUANTITY

PICTURE 9(5) USAGE IS COMP-3.

03 COMMISSION

PICTURE 9(5)V99.

01 INAREA.

02 INDICATOR-AREA-IN.

03 IND-30

PICTURE 9.

02 DATA-REGION-IN.

02 SALESMAN

PICTURE 9(20).

02 ITEM

PICTURE A(7).

02 QUANTITY

PICTURE 9(5) USAGE IS COMP-3.

PROCEDURE DIVISION.

.

.

.

MOVE 1 TO IND-10. (2)

DISPLAY OUTAREA UPON SCREENS USING SCREEN01. (3)

.

.

.

MOVE 1 TO IND-20. (4)

DISPLAY IND-10,IND-20,ITEM,QUANTITY,COMMISSION UPON SCREENS USING SCREEN01.

.

.

.

ACCEPT INAREA FROM SCREENS USING SCREEN01. (5)

.

.

.

- ① Here, the indicators are grouped with the output data fields. Since the format was created with indicators, the SFC knows that the first several bytes of OUTAREA are going to be indicator fields, not data fields.
- ② A simple MOVE statement is one way of turning an indicator on or off. An indicator is on when it contains the value of 1 and it is off when the value is 0. Although not shown in the example, you must remember to set an indicator value back to 0 to turn an indicator off.
- ③ Here the DISPLAY verb uses the group name OUTAREA, thus passing both the indicator region and the data region to the SFC. Only the data fields appear on the workstation operator's screen.
- ④ Here, instead of using the group name OUTAREA, the elementary items are listed in the DISPLAY statement. The result, however, is the same. The rules discussed in 7.3.2.2 regarding the DISPLAY verb also apply when indicators are used.
- ⑤ The field for the response indicator IND-30 is included in the ACCEPT statement. If an operator presses the user-assigned function key associated with that indicator, a value of 1 is passed back to the indicator field, setting the indicator on.

Figure 7-6. COBOL Program Entries for Indicators (Part 2 of 2)

7.3.2.3. Additional Coding Considerations for Function Keys and Multivolume Workstations

As noted previously, the SPECIAL NAMES paragraph and the ACCEPT and DISPLAY verbs have additional clauses not yet discussed. These additional clauses report function key input to a COBOL program and support multivolume workstations (where several terminals operate under the same lfd name).

The complete statement for the SPECIAL NAMES paragraph is:

```
SYSFORMAT IS mnemonic-name ASSIGN TO workstation lfd-name  
[CONTROL AREA IS record-name] [WITH FUNCTION-KEYS] [WITH CONNECT-FREE]
```

The CONTROL AREA clause specifies a 40-character area that receives data describing workstation activity. Among the benefits of having a control area are:

- Providing a better means of recovery when an error occurs. This is especially useful in a multiple-workstation environment. If your program encounters a problem with one workstation, it can be directed to handle the problem without affecting the other workstations connected to it.
- Reporting when a workstation is connected to or freed from your program and directing your program to follow a course of action as a result.
- Reporting function/command key input. You can direct your program to perform some operation when a workstation operator presses a user-assigned function/command key.

The control area can be defined in the WORKING-STORAGE or LINKAGE sections of your program. You code the control area as follows:

```

01 WRKSTN-CONTROL .
   05 WS-ID          PIC 999.
   05 FILLER         PIC X.
   05 WS-STATUS     PIC XX.
   05 FUNCTION-KEY  PIC 99.
   05 FORMAT-NAME   PIC X(8).
   05 NUMBER-CONNECTED PIC 99.
   05 SIZE-OF-DATA TRANSFER PIC 9(5).
   05 FILLER        PIC X(17).

```

- WRKSTN-CONTROL is a sample record name for the control area.
- WS-ID identifies the workstation where the most recent ACCEPT or DISPLAY operation was performed.
- WS-STATUS reports a 2-character status code for the workstation that performed the most recent ACCEPT or DISPLAY. Status code details are listed in Table 7-1.
- FORMAT-NAME is the name of the screen format that is active on the workstation terminal that was the object of the most recent ACCEPT or DISPLAY.

Table 7-1. Status Keys

| Status Key 1 | Status Key 2 |
|-------------------------|---|
| 0 Successful completion | 0 No further information |
| 1 At end | 0 Function key 15 received |
| | 6 Only active terminal has disconnected (Status key 98 may take precedence.) |
| 2 Invalid format | 3 Format not found |
| | 4 Format constructed incorrectly |
| 3 Permanent error | 0 No further information |
| 9 Workstation exception | 1 Terminal not compatible with format |
| | 2 Statement not compatible with format |
| | 3 Data not compatible with format |
| | 4 Data area not large enough for format |
| | 5 Function key, no data |
| | 7 New device connected, no data |
| | 8 Device disconnected (freed), no data |
| 9 Device not connected | |

- FUNCTION-KEY holds the number (from 1 to 22) of the function key pressed by an operator prior to the most recent ACCEPT. It is set to zero instead of a function key whenever data is accepted. The FUNCTION-KEY field is maintained only if the WITH FUNCTION-KEYS phrase is specified. You can have one response indicator set by more than one function key. In fact, any combination of function keys and response indicators may be specified.
- SIZE-OF-DATA-TRANSFER holds the number of characters actually sent to or received from the workstation screen. On a DISPLAY or ACCEPT operation, this number reflects the number of characters required by the format.

The WITH FUNCTION-KEYS clause specifies that whenever function key input is received on an ACCEPT verb, that function key value is reported to the control area.

The WITH CONNECT-FREE clause specifies that your program is directed to a particular procedure (an EXCEPTION path that is described later) when a terminal connects to or disconnects from your program.

The complete format for the ACCEPT statement is:

```
ACCEPT identifier-1 [,identifier-2...]
FROM SPECIFIC mnemonic-name
  [USING { identifier-3 }
   { literal } ]
[ON EXCEPTION imperative-statement]
```

SPECIFIC has meaning only for multivolume workstations. It indicates that input is to be accepted only from the particular workstation whose ID is given in WS-ID. If you use SPECIFIC, your program waits until the specified workstation supplies data, effectively locking out the other workstations.

You should avoid including a USING clause in an ACCEPT statement when you're working with multivolume workstation files. Under such conditions, the object of the USING clause in an ACCEPT statement is always the workstation that performed the most recent ACCEPT or DISPLAY (the workstation identified by WS-ID in the control area). This means if you specify a particular format name in the USING clause, that format will appear on only one workstation – the one that performed the last DISPLAY or ACCEPT. As a result, the workstation (and corresponding format) supplying the data for the ACCEPT statement may be different from the format (and corresponding workstation) specified by the USING clause.

The ON EXCEPTION clause directs your program to follow a different path whenever certain conditions arise, such as:

- When certain errors occur
- If a workstation is connected to or freed from your program
- When a user-assigned function key is pressed (This does not include function keys associated with a response indicator.)

If you specify a control area, you must include the ON EXCEPTION clause in both the ACCEPT and DISPLAY statements. If you do not specify a control area, you cannot use the ON EXCEPTION clause.

The ON EXCEPTION clause lets your COBOL program diagnose and recover from errors that occur as a result of communication between the program and the workstation. When an error occurs, the nature of the problem is reported in the WS-STATUS field of the control area as a 2-digit status code. Your program is directed to follow an exception path that you define. For example, consider the following:

```
ACCEPT INAREA FROM SCREENS USING SCREEN01
ON EXCEPTION PERFORM DIAGNOSIS-ROUTINE.
```

If an operator enters data that is not compatible with the active screen format, the status code 93 (Table 7-1) is reported to the WS-STATUS field of the control area. The ON EXCEPTION path is activated and the DIAGNOSIS-ROUTINE is performed. The DIAGNOSIS-ROUTINE could prompt some corrective action to be taken, as follows:

```
DIAGNOSIS-ROUTINE.
  IF WS-STATUS = 93
    PERFORM BAIL-OUT.
```

You can code procedures for the various status codes listed in Table 7-1, including procedures to follow when a workstation is connected to or freed from your program. As long as you include the CONNECT-FREE clause in the SYSFORMAT statement, each time a workstation connects to or is freed from your program, a status code is reported in the WS-STATUS field and an exception path is activated. Table 7-2 summarizes the effects of connect/free reporting.

Table 7-2. Effects of Connect-Free Reporting

| Workstation Options | Response to Connect or Free |
|-----------------------------------|--|
| CONTROL AREA WITH CONNECT/FREE | Set WS-STATUS to 97 or 98. Set WS-ID to report the device that connected or freed. Update NUMBER-CONNECTED. Execute the EXCEPTION clause. |
| CONTROL AREA without CONNECT/FREE | Update NUMBER-CONNECTED. If NUMBER-CONNECTED = 0, set status for end-of-file and execute the EXCEPTION clause. Otherwise, do not return control to the COBOL program until data is received. |
| CONTROL AREA not specified | If no terminal remains connected, terminate the program abnormally. Otherwise, do not return control to the COBOL program until data is received. |

A user-assigned function key can also activate an exception path as long as no response indicator is associated with it. When an operator presses a user-assigned function key, the function key value is reported in the FUNCTION-KEY field, status code 95 is returned to the WS-STATUS field, and the ON EXCEPTION clause is activated.

For example, suppose you want your own help screen displayed whenever function key 5 is pressed. As long as you've specified a CONTROL AREA and the WITH FUNCTION-KEYS clause, you can code your program as follows:

```
ACCEPT INAREA FROM SCREENS USING SCREEN01
ON EXCEPTION PERFORM DIAGNOSIS-ROUTINE
.
.
.
DIAGNOSIS-ROUTINE.
  IF WS-STATUS = 95 PERFORM PROCESS-FUNCTION-KEY.
.
.
.
PROCESS-FUNCTION-KEY.
  IF FUNCTION-KEY = 5
  PERFORM DISPLAY-HELP
```

If a function key is associated with a response indicator, the exception path is not activated. Only the function key number is returned to the FUNCTION-KEYS field of the control area. You must code your program to test for the value returned in the response indicator field and then direct your program to perform a specific function.

Table 7-3 summarizes the effects of function key input.

Table 7-3. Effects of Function Key Reporting

| Workstation Options | Response to Function Key Input | |
|---|---|--|
| | Response Indicator Set by Function Key | Response Indicators Absent or Unaffected |
| CONTROL AREA with WITH FUNCTION-KEYS | Return indicators without screen data. Set FUNCTION-KEY. Do not activate the EXCEPTION clause. | Set FUNCTION-KEY. Activate the EXCEPTION clause. |
| CONTROL AREA without WITH FUNCTION-KEYS | Return indicators without screen data. Do not set FUNCTION-KEY. Do not activate the EXCEPTION clause. | Ignore the function key input. Do not return control to the COBOL program until data is received. |
| CONTROL AREA not specified | Return indicators without screen data. | Ignore the function key input. Do not return control to the COBOL program until data is received. |

The complete format for the DISPLAY verb is:

```
DISPLAY identifier-1 [identifier-2]...
  UPON mnemonic-name
  [ USING { identifier-3 }
    { literal } ]
  [ ON EXCEPTION imperative-statement ]
```

The EXCEPTION clause for the DISPLAY statement performs most of the same functions as it does for the ACCEPT statement. It directs your program to follow an alternate path if certain errors occur when data is sent out to a screen format at a particular workstation.

Refer to the 1974 ANSI COBOL programmer reference for more information concerning COBOL programming in general.

7.3.2.4. Considerations for Workstations with Interprogram Communication

When you combine separately compiled COBOL programs to form a single run unit and to interact with the same workstation, these guidelines apply:

- In all programs, the SYSFORMAT IS statements should be identical except for the mnemonic-name, which is local to each program. While most workstation functions will still behave correctly if the SYSFORMAT statements aren't identical, function key input and connect-free events may occur without being reported to the COBOL program.
- You should establish a CONTROL AREA in the main program, reference it in the LINKAGE SECTION of the subprograms, and include it as an operand in the USING clause in CALL statements.

7.3.3. FORTRAN

Screen format services is activated by the FORTRAN system whenever the SCREEN=name option occurs in the sequential READ or WRITE statement as shown in the following examples:

```
1. READ (20,100,SCREEN='TESTSCR1')I
2. REAL*8 NAME/'TESTSCR2'/
   .
   .
   .
   WRITE (20,200,SCREEN=NAME)
```

The unit argument must be connected to a workstation device, and the screen name specified by the literal or variable after 'SCREEN=' must be the 1- to 8-character name that was assigned to the format (via the home screen) when it was created.

In cases where a format is used for both input and output, the screen name must be specified in the first WRITE statement to create the output fields. The same screen name must then be specified in a READ statement.

Both formatted and unformatted I/O is permitted for a screen format.

■ Formatted I/O

Each screen field must be defined with DISPLAY. The field lengths must exactly match the lengths of the actual screen format fields as specified during the SFG session. The following example requires that 'TESTSCR1' have a 3-character input display field:

```
      READ (20,100,SCREEN='TESTSCR1')I
      100 FORMAT (I3)
```

■ Unformatted I/O

Unformatted I/O is only permitted for characters and integers. Logical and real values are restricted. The internal lengths defined for the screen format must match the FORTRAN data sizes.

| <u>Data</u> | <u>Size</u> |
|-------------|-------------|
| INTEGER | 4 |
| REAL | 4 |
| REAL*8 | 8 |
| LOGICAL | 4 |
| INTEGER*2 | 2 |
| LOGICAL*1 | 1 |
| COMPLEX | 8 |
| COMPLEX*16 | 16 |

Refer to the FORTRAN IV programmer reference for more information concerning FORTRAN IV programming in general.

7.3.4. BAL

The imperative macroinstructions associated with using screen format services are OPEN, CLOSE, DMSEL (SELECT), DMINP (INPUT), and DMOUT (OUTPUT).

7.3.4.1. Open a File (OPEN)

You use this macroinstruction to initialize a file. This instruction establishes a logical access path (LAP) to the physical file. Only one file can be opened when you issue this instruction. If you want to access more than one file, you must issue an OPEN instruction for each of the files. You must issue this instruction for a file before you attempt any processing.

Format:

| LABEL | ΔOPERATIONΔ | OPERAND |
|--------|-------------|--|
| [name] | OPEN | {(cdibname) [(ribname)] {(1) } { (0) 1 } { 0 } |

Positional Parameter 1:

cdibname

Is the symbolic address of the CDIB that contains the screen name that was assigned to the file by the LFD job control statement.

(1) or 1

Indicates that you have preloaded register 1 with the address of the CDIB for the file to be opened.

Positional Parameter 2:

ribname

Is the symbolic address of the RIB associated with the file to be opened.

(0) or 0

Indicates that you have preloaded register 0 with the address of the RIB associated with the file. If you load register 0 with a value of zero, this indicates that there is no RIB associated with the file. (This also means that a RIB cannot reside at relative address X'000000'.)

If a symbolic address is specified for both positional parameter 1 and positional parameter 2, the comma shown in the instruction format is optional.

Example:

```
OPEN FILE1 RIB1
```

7.3.4.2. Close a File (CLOSE)

You use this macroinstruction to terminate file processing. Once a file is closed, it cannot be accessed until it is reopened by issuing an OPEN macroinstruction. Only one file can be closed when you issue this instruction. If you have more than one file open in your program, you must issue a CLOSE instruction for each of the files.

Format:

| LABEL | △OPERATION△ | OPERAND |
|--------|-------------|--------------------------|
| [name] | CLOSE | { cdibname (1) 1 } |

Positional Parameter 1:

cdibname

Is the symbolic address of the CDIB that contains the screen name that was assigned to the file by the LFD job control statement.

(1) or 1

Indicates that you have preloaded register 1 with the address of the CDIB associated with the file to be closed.

Example:

```
1. CLOSE SCREEN1
2. CLOSE 1
```

1. The file whose CDIB has the symbolic address SCREEN1 is to be closed.
2. The file whose CDIB address has been preloaded into register 1 is to be closed.

7.3.4.3. Retrieve a Record (DMINP)

The DMINP imperative makes data from a device available to a user program within a work area. The work area must be large enough to handle all input that has been entered by the operator. If function keys entered set response indicators, the appropriate number of response indicators must precede the data in the work area.

Format:

| LABEL | △OPERATION△ | OPERAND |
|--------|-------------|---|
| [name] | DMINP | {cdibname}, {work area} {(0)} {0} |

Positional Parameter 1:

cdibname

Specifies the logical file name that symbolically identifies the CBIB to be affected.

(1) or 1

Indicates that register 1 contains the address of the CDIB.

Positional Parameter 2:

work area

Specifies the symbolic address of a work area that is to contain all response indicators and input data.

(0) or 0

Indicates that register 0 is the address of a work area.

NOTES:

1. You need not issue a DMSEL imperative prior to the first imperative if JCL specification was provided with an initial screen selection.
2. If you issue the DMINP imperative without issuing the DMOUT imperative, screen format services issues a DMOUT imperative for the format in effect (whether specific or from JCL). All input fields are displayed with the replenish value. If any output or bidirectional fields exist, however, an error is returned and the DMOUT is not issued.

3. *If the input record size specified for variable-length records is not large enough, an error is returned and no read operation is issued.*
4. *For variable-length records, the size of the variable data plus the number of response indicators plus 4 followed by two blanks is moved to the first two bytes of the work area.*

If the DMINP is unsuccessful, CD\$DDINF+11 contains one of the following error codes:

- X'04' Invalid input record size
- X'05' Output only screen format
- X'06' Output only device
- X'07' Invalid SFS imperative sequence
- X'10' Record level format error
- X'11' Imperative abnormally terminated
- X'12' Field level format error
- X'13' DSD field level format error
- X'14' Undecipherable characters in the input buffer
- X'15' Physical I/O error
- X'16' Conversion error
- X'17' Maximum retries exceeded
- X'18' DSD syntax error
- X'19' Function key 16 entered

7.3.4.4. Select a Screen Format (DMSEL,SCREEN)

You use the DMSEL,SCREEN imperative macroinstruction to indicate which screen format is to be displayed.

Format:

| LABEL | △OPERATION△ | OPERAND |
|--------|-------------|---|
| [name] | DMSEL | { cdibname }, SCREEN, { (1) 1 } { screen format name } { (0) 0 } |

Positional Parameter 1:

cdibname

Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

(1) or 1

Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2:

SCREEN

Indicates that you are selecting a screen format from a screen format file.

Positional Parameter 3:

screen format name

Is the symbolic address of an 8-byte area that contains the name of the screen format to be selected.

(0) or 0

Indicates that you have preloaded the address of the screen format name in register 0.

Example:

```
DMSEL WORKSTN,SCREEN,S1
```

Selects the screen format named in the 8-byte area S1.

If the format name provided contains all X'00', the effect is to terminate screen format services. If you subsequently issue a DMSEL with a format name, screen format services is reactivated.

NOTE:

Resetting screen format services is not permitted for a multiworkstation file. An invalid imperative error code is returned should an attempt be made to do so.

The error conditions returned in CD\$DDINF+11 are:

- X'21' Format name specified could not be found
- X'22' I/O error while reading the format
- X'20' Format width greater than screen width
- X'11' Imperative abnormally terminated

7.3.4.5. Output a Record (DMOUT)

The DMOUT imperative permits the user data to be merged from the user's work area with a previously selected format, which is then presented to the device. The keyboard is also unlocked.

Format:

| LABEL | △OPERATION△ | OPERAND |
|--------|-------------|---|
| [name] | DMOUT | { cdibname }, { work area } { (1) } { (0) } 1 } 0 } |

Positional Parameter 1:

cdibname

Specifies the logical file name that symbolically identifies the CDIB to be affected.

(1) or 1

Indicates that register 1 contains the address of the CDIB.

Positional Parameter 2:

work area

Specifies a positional parameter defining a user work area that contains any option indicators and all fields to be merged with the format.

(0) or 0

Indicates register 0 points to the user's work area.

If the output record size specified for variable-length records is not large enough, an error will be returned and the partial buffer will not be sent.

The error codes reported in CD\$DDINF+11 are:

- X'03' Invalid output record size
- X'08' I/O or INT-2 conversion error
- X'09' Invalid screen format
- X'11' Imperative abnormally terminated
- X'26' Output verification error
- X'29' Device freed

7.3.5. ESCORT

When you create a screen format, you give ESCORT a special format in which to output file data. You still designate a structure, but instead of your file being output in the default structure format, it's linked with your screen format. When you designate a screen format, the fields you link to it must be identical to the fields contained in a structure. They must have the same name, and they must be in the same order as the corresponding fields in your structure. Both the format and your structure must also have the same number of fields.

For example, if you designate 10 fields in a format and link it to a structure with only 9 fields, you get an error because ESCORT keeps looking for the 10th field.

It is possible, however, to have a structure with 10 fields and an output format using only 5 fields, but only the first 5 fields of your structure are output. Although you can do it this way, it's always better to have a direct structure/format linkup to eliminate confusion.

Depending on the type of ESCORT program you use with a format (ENTER DATA, CHANGE DATA, SELECT DATA), the following requirements apply:

- ENTER DATA

Specify all fields in the format as input (I).

- CHANGE DATA

Specify all fields in the format as both input and output (B).

Protect the decimal point in decimal numbers with a slash.

You cannot use packed decimal fields in a format. The initial display does not display underlines for the packed field.

When the format is first displayed, only the key field is read. ESCORT ignores all other data until the master record is displayed and changed.

- SELECT (display) DATA

Specify all fields as both input and output (B) or as output only (O). When you display data using only output fields, you must define at least one input field to hold the screen until the XMIT key is pressed. ESCORT ignores any data sent from this input field (B or I).

7.3.6. IMS Programming Considerations

- General Considerations:

- UNISCOPE 100 and 200 terminals must have the PROTECT feature.
- UTS 400 terminals (if operating in native mode) must have the PROTECT feature; FCC switch set to FCC; and control page set to XMIT variable.
- SFS parameter specifying the number of terminals that use screen formats at the same time.
- RESFMT parameter indicating the maximum number of screen formats that remain resident between SFS calls.
- CALL 'BUILD' and CALL/ZG#CALL BUILD commands call screen format services in COBOL and BAL action programs, respectively. A K in column 42 of the output specifications forms calls screen format services in RPG II action programs. You must not specify the USE SFS job control statement.
- The IMS CALL REBUILD or the data management DMCTL INERR/INERRMSG imperatives may not be used in conjunction with formats containing conditional protect and/or hidden field. Instead, the program could use option indicators to blink the field via a CALL BUILD or DMOUT imperative.

- RPG II Action Program Considerations:

- Input Format Specifications Form

Every variable data field with an I/O direction of input or both input and output (bidirectional) must be identified on the input format specifications form. The field names need not correspond to those that were assigned to the actual format fields during generation.

The field location entries on the RPG II form refer to the position of these fields with respect to the input data rather than to the position of the input fields on the screen. This is simply because data is either read from the format (from input or bidirectional fields) or written to the format (to output or bidirectional fields) automatically – according to the sequence in which the variable data fields were described when the format was created.

Field lengths are counted sequentially. The FROM entry for the first input or bidirectional field is always 1, and the TO entry is always the number of the last underline in the field as defined in the format. The FROM entry for the next field picks up where the TO entry for the previous field left off. The underlines are counted in relation to this number to get the TO entry.

Let's say that the first input or bidirectional field in the format has a length of 10, and the second has a length of 12. The FROM and TO entries for the first field are 1 and 10. Entries for the second field are therefore 11 and 22. Output fields are not counted at all. If an output field is located between two input or bidirectional fields, simply bypass it.

Control levels, matching fields, and look-ahead fields may not be used with workstation input.

– Output Format Specifications Form

Every variable data field with an I/O direction of output or bidirectional must be identified on the output format specifications form. This includes any bidirectional field that was defined on the input format specifications form.

The END POSITION entries refer to the end positions within the output data and not to their positions on the screen. For example, if there are two output or bidirectional fields with lengths of 10 and 12 respectively, the end position for the first field is 10, and that of the second field is 12.

The name of the format, as it was specified on the home screen, must be entered in the CONSTANT OR EDIT WORD field (columns 45–52) and enclosed in apostrophes. The length of the format name must be entered in column 43 with the letter K preceding it (column 42).

Use of a first-page indicator is not permitted.

For more details regarding RPG II action program considerations, refer to IMS action programming in RPG II, UP-9206 (current version).

■ COBOL Action Programs

The COBOL compiler does not interface with screen format services. All interfaces with screen format services are through IMS (CALL BUILD). See Appendix F for a sample IMS COBOL action program (JAMENU) for screen format services and refer to IMS action programming in COBOL and Basic Assembly Language (BAL), UP-9207 (current version), for a detailed description.

- BAL Action Programs

Same as COBOL

7.4. CONSIDERATIONS FOR THE WORKSTATION OPERATOR

The following guidelines apply when you enter data on a screen format:

- After you enter data on a screen, make sure the cursor is beyond the last field of the format before transmitting. If you return to a particular field *to correct an error before transmission*, correct the error, position the cursor beyond the last field, and then transmit.
- If you return to a particular field *to correct an error after transmission*, simply correct the error and transmit. You don't have to position the cursor beyond the last field.
- The following are function keys you should not use and operations you should not perform while a screen format is active. Disruption of device control information in the workstation and possibly termination of input transmission may occur. (An SF16 error results.) The restricted keys and operations are:
 - INSERT LINE
 - DELETE LINE
 - LINE DUP
 - DISPLAY ERASE
 - FCC GEN
 - FCC CLEAR
 - FCC ENABLE
 - RESET
 - TAB SET (causes an SF14 error)
 - Changing the control page XMIT option to other than VAR
 - Switching to simulated system mode on a UNISCOPE terminal
 - Powering off the workstation

Appendix A. Job Control Considerations

A.1. THE USE SFS STATEMENT

You must indicate within the DVC/LFD sequence for the workstation that screen formats are to be used. This is done via the USE SFS job control statement. The optional positional parameters in this statement may be used as necessary. The format for the statement is:

```

//[symbol] USE SFS [ , [[format-file-lfd-1]/[format-file-lfd-2] ] ]
                  [ { SYSTEM
                    [format-file-lfd
                  ] } ]
[,initial-screen] [ { nnn
                    [ 1 ] } ]
[,screen-format-1=alias-1,...,screen-format-12=alias-12]

```

where:

SFS

Indicates that the program is to use a screen format.

```

[ , { [[format-file-lfd-1]/[format-file-lfd-2] ] ]
    [ { SYSTEM
      [format-file-lfd
    ] } ]

```

In this parameter, you can provide an lfd name for up two screen format files. Any name you use must match an lfd name specified in a previously defined device assignment set for a screen format file. (Screen format files are always MIRAM files.) The format-file-lfd is one to eight alphanumeric characters long.

When coding this parameter, remember the following:

- If you omit a format-file-lfd name, it is assumed that all screen formats used reside in the system file \$Y\$FMT.
- If you code /format-file-lfd-2 alone, \$Y\$FMT is examined first then the file indicated by format-file-lfd-2.

- If you code `format-file-lfd-1/` alone, the file indicated by `format-file-lfd-1` is examined first then `$$FMT`.
- If you code `format-file-lfd` alone, only the file indicated by `format-file-lfd` is examined.
- If you specify `format-file-lfd-1/format-file-lfd-2`, a search will be made in `format-file-lfd-1` first. If the format isn't found, `format-file-lfd-2` will be searched.

`initial-screen`

The initial screen name refers to the 1- to 8-character name (as defined on the home screen) for a format. Use of this parameter depends on the program's language.

BAL

The initial screen name indicates the first format to be used by the program. A `DMSSEL` imperative need not be issued to use this format as the initial format. Refer to 7.3.4 and the consolidated data management macro language user guide/programmer reference for further information on BAL macros.

COBOL

The initial screen name indicates the first format to be used by the program.

Refer to 7.3.2 and the 1974 ANSI COBOL programmer reference for further information on the `SYSFORMAT IS` statement and the use of screen formats with a COBOL program.

RPG II

This parameter should not be used with an RPG II program. Refer to 7.3.1 and the RPG II user guide for further discussion of RPG II programming and screen formats.

FORTTRAN IV

This parameter may be used with a FORTRAN program. Refer to 7.3.3 and the FORTRAN IV programmer reference for more information on FORTRAN programming.

`{nnn}`
`{ 1 }`

This parameter indicates the number of formats to be present in main storage at any one time. It should only be used if the program is going to alternate regularly between two or more formats. The default for this parameter is 1, but you may specify any number from 2 to 255. The number of formats that the program uses is not limited to the number specified here.

If you accept the default, regardless of how many formats a program is going to use, only one format will ever be present in main storage. When the first format is called by the program, it is placed in main storage. When the second is called, it replaces the first, and so on.

The number 3 entered for this parameter indicates that three formats can reside in main storage throughout the program use. You might use this parameter when a program is constantly going to alternate between the same three formats. As a result, the I/O activity that would be necessary to bring the formats back and forth (if the default were accepted) is eliminated. By the same token, however, more main storage is used up to accommodate three formats.

Remember, the number 3 does not restrict the program to the use of only three formats. It means that three formats can reside in main storage at once. If the program uses only three, the same three will always be present. If it uses more, which three formats are present depends on which ones the program calls and when.

screen-format-1=alias-1,...

This parameter permits you to equate the name that was given to the format when it was created with a name that the program uses for that format.

Suppose SCREEN01 was the name specified on the home screen during generation. Supplying the alias parameter as SCREEN01=MYSCREEN means that the program is using the name MYSCREEN instead of SCREEN01. No more than 12 alias names may be specified.

The following is an example of the USE statement with some accompanying parameters.

```
// USE SFS,FORMFIL,,2,SCREEN020MYSCREEN
```

This statement indicates:

- The formats to be used are on a library file other than \$Y\$FMT. The lfd-name for that file is FORMFIL.
- Two formats will reside in main storage.
- The program will use the name MYSCREEN when referencing SCREEN02.

The following are some examples of job control DVC/LFD sequences containing the USE SFS statement.

Example 1:

```

// JOB PAYROL
.
.
.
// DVC 200      } DVC/LFD sequence
// USE SFS      } for the workstation
// UID $Y$MAS   } file
// LFD WRKSTN  }
.
.
.
// EXEC PRSNL

```

Example 2:

```

// JOB PAYROL
.
.
.
// DVC 200      } DVC/LFD
// USE SFS,,,,,SCREEN020MYSCREEN } sequence
// UID W1       } for the
// LFD WRKSTN   } workstation
.               } file
.
.
.
// EXEC PRSNL

```

Both of these examples contain a DVC/LFD sequence for the workstation where the device code number is 200 and the name that the program will use to reference the workstation is WRKSTN. In both cases, the formats to be used are on \$Y\$FMT; therefore, no LFD/DVC sequence is given for this file and the USE statement does not specify a format's file lfd name. The USE statement in example 2 specifies the alias parameter.

In both examples, the formats are stored on a user-assigned MIRAM library file; consequently, a DVC/LFD sequence for this file must be given. The USE SFS statement within the workstation DVC/LFD sequence must specify the lfd-name for the user library file as shown. The number of formats to be present in main storage at any one time is specified as 3. The alias parameter is also specified. For more information concerning job control in general, refer to the job control user guide.

If your program interfaces with both the Menu Processor and Screen Format Services you need only supply the // USE statement specifying the MENU option. Refer to menu services concepts and facilities, UP-9317 (current version). The SFS option for the // USE statement is not necessary for the dual interface, provided you perform screen selection within the program.

A.2. JOB CONTROL FOR MULTIPLE WORKSTATIONS

When building a job control stream for a job that uses screen formats, you may specify the use of more than one workstation. This allows several operators to work with a screen format at once. The following is an example of a job control stream that specifies multiple workstation use.

| | |
|---|--|
| Request for EDT (the general editor) | LOGON MG2 EDT |
| Device assignment set (DVC/LFD sequence) for the workstation | } // JOB SCREENIT // DVC 200(3),REQ(3) // USE SFS // UID W1,W2,W3 // LFD WORKSTN |
| Device assignment set for the payroll program | } // DVC 51 // VOL D000028 // LBL LODLIBRARY // LFD LODLIB |
| Device assignment set for the input data file | } // DVC 52 // VOL D000028 // LBL INPAYROLL // LFD INPAY |
| Device assignment set for the output data file | } // DVC 52 // VOL D000028 // LBL OUTPAYROLL // LFD OUTPAY |
| | // EXEC PAYROLL,LODLIB /& |
| EDT commands | } @WRITE MODULE=SCREENIT,FILENAME=SAVEJOBS,VSN=D000028 @HALT LOGOFF |

The DVC for the workstation indicates that three workstations are to be used and that the job will not be scheduled until three workstations are available (as indicated by the REQ parameter).

Although not directly related to the DVC and UID statements for multiple workstation use, there are some other items related to this job control stream worth mentioning.

- When working with multiple workstations, you may include the FREE statement either within the job control stream or enter it directly at the workstation as a command to free one or more workstations when their use is no longer required.
- As shown in the example, you may use the EDT, @WRITE, and @HALT commands so that the next time you want to run this job all you have to do is enter

```
LOGON MG2  
RV SCREENIT,(SAVEJOBS,D000028)
```

at the workstation.

- You must include the USE SFS statement (A.1) in the DVC/LFD sequence for the workstation whenever screen formats are used.
- Since there is no DVC/LFD sequence for a screen format file, it is assumed that all formats are in the library \$Y\$FMT on SYSRES.

For more information about job control, see the job control user guide, UP-9986 (current version). For more information about EDT, see the general editor user guide/programmer reference, UP-9976 (current version).



Appendix B. The 12-Line Screen Display

The following screen formats should be used as a reference for operators that execute screen format services on a workstation with only a 12-line capacity. Katakana workstations display data in uppercase letters.

HOME SCREEN

```
FUNCTION (1): 1 CREATE    2 CREATE-FROM    3 MODIFY    4 DELETE
              5 SHOW      6 LIST      7 SPOOL     8 TERMINATE
OLD FORMAT NAME (_____) IS ON THE FOLLOWING LIBRARY:
FILE NAME: ($SFMT          ) VOLUME: (RES    )

NEW FORMAT NAME (_____) IS TO BE STORED ON THE FOLLOWING LIBRARY:
FILE NAME: ($SFMT          ) VOLUME: (RES    )
IF THIS FILE DOES NOT EXIST, ALLOCATE (002) CYLINDERS. INCREMENT IS (01) CYL.

**FUNCTION KEYS ARE:F1-GO TO HOME SCREEN, F5-BREAKPOINT SPOOL FILE,
                    F13-HELP, F14-EXIT HELP, F20-RESTORE SCREEN
```

CHARACTERISTICS SCREEN

```
GLOBAL CHARACTERISTICS FOR FORMAT xxxxxxxx:
ERROR RETRY COUNT:(02) ALPHABET:(ENGLISH) LOWER CASE TRANSLATION:(1) 1 YES 2 NO
SCREEN FORMAT IS (1): 1 ORIGINAL 2 OVERLAY
ERASE/UNLOCK BY (1): 1 NONE 2 REPLENISH 3 ERASE 4 UNLOCK KEYBOARD 5 CONDITIONAL
TRANSMIT ALL, DISREGARD CURSOR POSITION? (1): NO 2 YES
EDITING CHARACTERS? (1): 1 NO 2 YES SPECIAL DISPLAY CONTROL? (1): 1 NO 2 YES
DO YOU WISH TO SPECIFY AN ERROR MESSAGE FIELD? (1): 1 NO 2 YES
DISPLAY RETENTION ON ALL FIELDS? (1): 1 NO 2 YES
FUNCTION COMMAND KEYS TO BE DEFINED? (1): 1 NO 2 YES
DOES THIS FORMAT HAVE A NON-DISPLAYED CONSTANT? (1): 1 NO 2 YES
```

CONDITIONAL ERASE/REPLENISH SCREEN

ENTER THE CONDITIONAL ACTION DESIRED AFTER INPUT FOR FORMAT nnnnnnnn

DO YOU WISH ALL INPUT-ONLY FIELDS TO BE REPLENISHED AFTER INPUT
BASED UPON A CONDITIONAL INDICATOR? (1): 1 NO 2 YES INDICATOR (Y ___)

DO YOU WISH TO HAVE THE SCREEN ERASED AFTER INPUT
BASED UPON A CONDITIONAL INDICATOR? (1): 1 NO 2 YES INDICATOR (Y ___)

EDIT SCREEN

ENTER EDIT CHARACTERS IN PARENTHESES TO BE USED FOR FORMAT nnnnnnnn

| STANDARD CHAR | MEANING: | = CHAR TO BE USED |
|---------------|--------------------|-------------------|
| \$ | CURRENCY SIGN | =(\$) |
| . | DECIMAL SYMBOL | =(.) |
| , | THOUSANDS PUNCT. | =(,) |
| * | REPLACEMENT CHAR | =(*) |
| CR | CREDIT ON NEGATIVE | =(CR) |
| DB | DEBIT ON NEGATIVE | =(DB) |
| _ | REPLENISH CHAR | =(_) |

NOTE: MAKE SURE NO SYMBOL IS AMBIGUOUS WITH ANY VALID PICTURE STRING CHARACTER.

SPECIAL DISPLAY SCREEN

ENTER THE DISPLAY PROPERTIES FOR FORMAT nnnnnnnn:

| | INTENSITY | EMPHASIS: |
|-------------------------|-----------|-----------|
| DISPLAY CONSTANTS | (B) | (____) |
| OUTPUT VARIABLES | (B) | (____) |
| INPUT CAPABLE FIELDS | (A) | (____) |
| FIELDS ENTERED IN ERROR | (A) | (5____) |

INTENSITY:

A. NORMAL INTENSITY
B. ALTERNATE (LOW INTENSITY)

EMPHASIS:

1. UNDERLINED 4. REVERSE VIDEO
2. COLUMN SEPARATORS 5. BLINK FIELD
3. STRIKE THRU BARS

ERROR MESSAGE SCREEN

ENTER THE FOLLOWING INFORMATION FOR YOUR ERROR MESSAGES FOR FORMAT nnnnnnnn
 MESSAGE FIELD NAME IS: (ERRORMSG)
 NUMBER OF LINES IN ERROR MESSAGE: (1) ENTER 1 OR 2
 DISPLAY ERROR MESSAGE ON LINE NUMBER: (LAST) ENTER NUMBER OR 'LAST'
 ERROR MESSAGES CONDITIONALLY DISPLAYED: INDICATOR (Y ___)
 DISPLAY ATTRIBUTES: INTENSITY (A) EMPHASIS: (____)

INTENSITY: EMPHASIS:
 A. NORMAL INTENSITY 1. UNDERLINED 4. REVERSE VIDEO
 B. ALTERNATE (LOW) INTENSITY 2. COLUMN SEPARATORS 5. BLINK FIELD
 3. STRIKE THRU BARS

DISPLAY RETENTION SCREEN

DO YOU WISH THE DISPLAY IMAGE TO BE CONDITIONALLY RETAINED
 FOR ALL FIELDS OF FORMAT nnnnnnnn? (1): 1 NO 2 YES

INDICATOR (Y ___)

(IF THIS IS SELECTED, AN OUTPUT COMMAND WILL AFFECT DISPLAY ATTRIBUTES ONLY
 UNLESS OVERRIDDEN AT FIELD LEVEL VIA DIALOG 5.)

FUNCTION/COMMAND KEY SCREEN

ENTER THE FUNCTION/COMMAND KEYS TO BE USED BY FORMAT nnnnnnnn

| KEY# | ACCEPT? | RESPONSE | KEY# | ACCEPT? | RESPONSE | KEY# | ACCEPT? | RESPONSE |
|------|---------|----------|------|---------|----------|------|---------|----------|
| F1 | (N) | (___) | F2 | (N) | (___) | F3 | (N) | (___) |
| F4 | (N) | (___) | F5 | (N) | (___) | F6 | (N) | (___) |
| F7 | (N) | (___) | F8 | (N) | (___) | F9 | (N) | (___) |
| F10 | (N) | (___) | F11 | (N) | (___) | F12 | (N) | (___) |
| F13 | (N) | (___) | F14 | (N) | (___) | F15 | (N) | (___) |
| F16 | (N) | (___) | F17 | (N) | (___) | F18 | (N) | (___) |
| F19 | (N) | (___) | F20 | (N) | (___) | F21 | (N) | (___) |
| F22 | (N) | (___) | | | | | | |

(USE FUNCTION KEY#13 FOR HELP. THIS WILL EXPLAIN INDICATOR SYNTAX.)

NONDISPLAY SCREEN

DESCRIBE THE SINGLE NON-DISPLAYED FIELD FOR FORMAT nnnnnnnn

LENGTH OF THIS FIELD IS (__)

VALUE OF FIELD IS (1):

- 1 - FIRST FIELD OUTPUT BY PROGRAM (FIELD IS BIDIRECTIONAL)
- 2 - FIELD IS INPUT-ONLY, VALUE IS:

THIS VALUE IS ALWAYS THE FIRST INPUT FIELD WHICH IS NAMED: (FLD000000)
IF OPTION #1 IS SELECTED ABOVE, IT IS ALSO THE FIRST OUTPUT FIELD.

DIALOG SCREEN 1

display of the field

(FLDnnnnn) IS FOR FIELD USE OF (1): 1 OUTPUT 2 INPUT 3 BOTH
INTERNAL USAGE IS (1): 1 DISPLAY 2 PACKED 3 BINARY 6 ZONED
INTERNAL LENGTH IS (nn)

PLEASE INDICATE WHETHER THE FOLLOWING ARE REQUIRED: 1 NO 2 YES

- | | | | |
|---------------------------|-----|----------------------------|-----|
| CONDITIONAL DISPLAY | (1) | SPECIAL DISPLAY PROPERTIES | () |
| CONDITIONAL RETENTION | (1) | CONDITIONAL PROTECTION | () |
| FIELD CHANGE NOTIFICATION | (1) | RANGE CHECKING | () |

REPLACE "!" IN THE FOLLOWING LINE WITH INSERTION CHARACTERS:

display of the field

DIALOG SCREEN 2

1. SOCIAL SECURITY: 999-99-9999
- 2.
3. Field response (1): 1 OPTIONAL 2 REQUIRED 3 MUST-FILL 4 DEFAULT VALUE IS:
4. -----
- 5.
6. Lowercase translation (1): 1 Yes 2 No
- 7.
- 8.
9. Initial replenishing value is:
10. -----
11. NOTE: For DEFAULT, alpha characters will be passed to program in upper or
12. lower case as entered. For REPLENISH, alpha characters will be displayed
13. as entered.

DIALOG SCREEN 3*display of field*

SPECIFY THE CONDITIONAL INDICATOR WHICH IS TO CONTROL DISPLAY OF FIELD FLDnnnnn
(Y ___)

THE FIRST CHARACTER SHOULD BE 'Y' IF DISPLAY IS TO OCCUR WHEN THE INDICATOR IS ON OR 'N' IF DISPLAY IS TO OCCUR WHEN INDICATOR IS OFF.

F-KEY 13 WILL PROVIDE HELP

DIALOG SCREEN 4*display of field*

| | | |
|--|------------|-----------|
| DISPLAY PROPERTIES FOR FIELD FLDnnnnn: | INTENSITY: | EMPHASIS: |
| LOWEST PRECEDENCE: I.DEFAULT PROPERTIES | (A) | (_____) |
| II.OPTION IND (_ ___) | () | (_____) |
| III.OPTION IND (_ ___) | () | (_____) |
| IV.OPTION IND (_ ___) | () | (_____) |
| HIGHEST PRECEDENCE:V.OPTION IND (_ ___) | () | (_____) |

INTENSITY:

EMPHASIS:

| | | |
|-----------------------------|---------------------|-------------------|
| A.NORMAL INTENSITY | 1.UNDERLINED | 4.REVERSE VIDEO |
| B.ALTERNATE (LOW) INTENSITY | 2.COLUMN SEPARATORS | 5.BLINK FIELD |
| C.NOT DISPLAYED | 3.STRIKE THRU BARS | 6.POSITION CURSOR |
| E.SAME AS FIELDS IN ERROR | | |

DIALOG SCREEN 5*display of field*

SPECIFY THE INDICATOR YOU WISH TO CONTROL DISPLAY RETENTION FOR FLDnnnnn
(Y ___)

(THIS INDICATOR WILL OVERRIDE FORMAT-LEVEL DISPLAY RETENTION FOR THIS FIELD, IF SPECIFIED. IF THIS CONDITION IS MET, AN OUTPUT COMMAND WILL AFFECT THE DISPLAY ATTRIBUTES ONLY.)

F-KEY 13 WILL PROVIDE HELP

DIALOG SCREEN 6*display of field*

SPECIFY THE INDICATOR YOU WISH TO CONTROL PROTECTION OF FIELD FLDnnnnn
(Y ___)

IF THE FIRST CHARACTER IS 'Y', THE FIELD WILL BE PROTECTED WHEN THE INDICATOR IS ON. IF 'N' IS SPECIFIED, PROTECTION OCCURS WHEN THE INDICATOR IS OFF.

F-KEY 13 WILL PROVIDE HELP

DIALOG SCREEN 7

SOCIAL SECURITY: 999-99-9999

SPECIFY THE INDICATOR YOU WISH TO CONTROL NOTIFICATION OF FIELD CHANGE FOR FIELD FLD00007:

INDICATOR VALUE: (___)

NOTE: THIS INDICATOR VALUE WILL BE SET (I.E., TURNED 'ON') IF THIS FIELD IS CHANGED ON INPUT.

FUNCTION KEY #13 WILL PROVIDE HELP.

DIALOG SCREEN 8

SOCIAL SECURITY: 999-99-9999

SPECIFY THE TYPE OF CHECK AND RANGE VALUE(S) FOR FIELD FLD00000:

RANGE CHECK (1): 1 BETWEEN 2 LESS THAN 3 GREATER THAN 4 EQUAL
5 NOT EQUAL 6 LESS/EQUAL 7 GREATER/EQUAL

RANGE VALUE(S):

MORE? (1): 1 NO 2 YES

MODIFY SCREEN

MODIFY OPTION (1): 1 CHANGE TEMPLATE 2 CHANGE TYPE 3 CHANGE I/O
7 CHANGE TEMPLATE ONLY - NO INTERNAL CHANGES

SUMMARY SCREEN

NAME: FORMAT01 FORMAT SIZE: 08 X 64 DATE: 82/10/29 CREATED FROM:
 ALPHA:ENGLISH FILE: \$Y\$FMT ON RES GEN8
 LINE ORIGIN: 01,01 OVERLAY: NO ERASE/UNLOCK BY: CONDITIONAL INDICATOR
 LOWER CASE TRANSLATION: YES NUMBER OF VARIABLES: 0014 ERROR RETRY COUNT: 02
 TRANSMIT ALL, REGARDLESS OF CURSOR POSITION: NO
 SPECIAL EDITING CHARACTERS: YES SPECIAL DISPLAY CONTROL: YES
 ERROR MSG. FIELD SPECIFIED: YES DISPLAY RETENTION: NO
 FUNCTION/COMMAND KEYS DEFINED: YES
 FORMAT HAS A NON-DISPLAY CONSTANT: NO

INPUT ONLY FIELDS TO BE REPLENISHED AFTER INPUT - INDICATOR IS Y 006

SPECIAL EDITING DISPLAY

NAME: FORMAT01 FORMAT SIZE: 07 X 64 DATE: 82/05/25
 SPECIAL EDITING CHARACTERS: YES

| STANDARD CHAR: | MEANING | = CHAR TO BE USED |
|----------------|--------------------|-------------------|
| \$: | CURRENCY SYMBOL | = (\$) |
| .: | DECIMAL POINT | = (/) |
| ,: | THOUSANDS PUNCT. | = (,) |
| *: | REPLACEMENT CHAR | = (*) |
| CR: | CREDIT ON NEGATIVE | = (CC) |
| DB: | DEBIT ON NEGATIVE | = (DD) |
| _: | REPLENISH CHAR. | = (_) |

DISPLAY CONTROL AND ERROR MESSAGE DISPLAY

NAME: FORMAT01 FORMAT SIZE: 07 X 64 DATE: 82/05/25
 DISPLAY CONSTANTS: (B)(____) OUTPUT VARIABLES :(B)(____)
 INPUT CAPABLE VARIABLES: (A)(____) FIELDS ENTERED IN ERROR:(A)(1____)
 ERROR MESSAGE TO BE DISPLAYED ON LINE NUMBER: LAST NUMBER OF LINES 1
 ERROR MSG. FIELD NAME IS: ERRORMSG ERR.MSG DISPLAY ATTRIBUTES:(B)(____)
 ERROR MESSAGE TO BE CONDITIONALLY DISPLAYED BASED ON INDICATOR (Y 004)

| INTENSITY: | EMPHASIS |
|-----------------------------|--------------------------------------|
| A.NORMAL INTENSITY | 1.UNDERLINED 4.REVERSE VIDEO |
| B.ALTERNATE (LOW) INTENSITY | 2.COLUMN SEPARATORS 5.BLINK FIELD |
| | 3.STRIKE THRU BARS |

LIST SCREEN

NAME: FORMAT01 FORMAT SIZE: 7 X 64 DATE: 82/25/05

| FIELD-ID | (X,Y) | I/O | EXTRNL | | INTRNL | | RESPN | REPLN | RANGE | COND |
|----------|-------|-----|--------|------|--------|-----|-------|-------|-------|------|
| | | | TYP | LNHG | TYP | BYT | | | | |
| FLD00001 | 1,55 | O | N | 2 | D | 2 | 2. | M | - | |
| FLD00002 | 1,58 | I | N | 2 | D | 2 | 2. | (Y) | (Y) | (Y) |
| NAME | 2,7 | I | A | 30 | D | 30 | 30. | R | - | |
| ADDR | 3,7 | B | X | 30 | D | 30 | 30. | R | - | |
| STATE | 3,46 | B | A | 4 | D | 4 | 4. | R | - | |
| FLD00006 | 3,56 | B | N | 5 | P | 3 | 5. | O | - | () |

MORE? Y OR N ()

SUBLIST SCREEN

| FIELD-ID | (Y,X) | I/O | EXTRNL | | INTRNL | | RESPN | REPLN | RANGE | COND |
|----------|-------|-----|--------|------|--------|-----|-------|-------|-------|------|
| | | | TYP | LNHG | TYP | BYT | | | | |
| FLD00002 | 1,58 | B | N | 2 | D | 2 | 2. | (Y) | (Y) | (Y) |

DEFAULT VALUE:
99

LOWER TO UPPERCASE TRANSLATION IN EFFECT.

RANGE:

- BTW 01-04
- BTW 06-08
- GT 13

CONDITIONAL VALUES SCREEN

| FIELD-ID | (X,Y) | I/O | EXTRNL | | INTRNL | | RESPN | REPLN | RANGE | COND |
|----------|-------|-----|--------|------|--------|-----|-------|-------|-------|------|
| | | | TYP | LNHG | TYP | BYT | | | | |
| FLD00006 | 03,56 | B | N | 05 | P | 03 | 05.00 | O | - | (Y) |

FIELD IS TO BE DISPLAYED IF INDICATOR 009 IS ON.
FIELD IS TO BE RETAINED.
FIELD IS TO BE PROTECTED IF INDICATOR 002 IS OFF.

*(LOWEST PRECEDENCE) - FIELD DEFAULT DISPLAY PROPERTIES:
INTENSITY: NORMAL EMPHASIS:

*(HIGHEST PRECEDENCE) - FIELD DISPLAY PROPERTIES:
INTENSITY: NORMAL EMPHASIS: R-VIDEO

WITH OPTION INDICATOR Y 014

Appendix C. Function Keys

The following function keys provide general assistance in using the SFG.

- Function key 1

Aborts processing of the current function and returns you to the HOME SCREEN.

- Function key 5

Causes the screen format generator to breakpoint the spool file to a printer. All records currently in the spooled printer output are directed to a printer (pending an available device). Records added to the spool file after the breakpoint are held until a subsequent breakpoint is requested or until the end of the job.

- Function key 13

Requests help for the user. The SFG will display the appropriate procedures and rules that apply to the particular stage of format generation with which you are involved. Press function key 14 to terminate the help screen or press the XMIT key to receive additional HELP screens.

- Function key 14

Returns you to normal processing after HELP is displayed.

- Function key 15

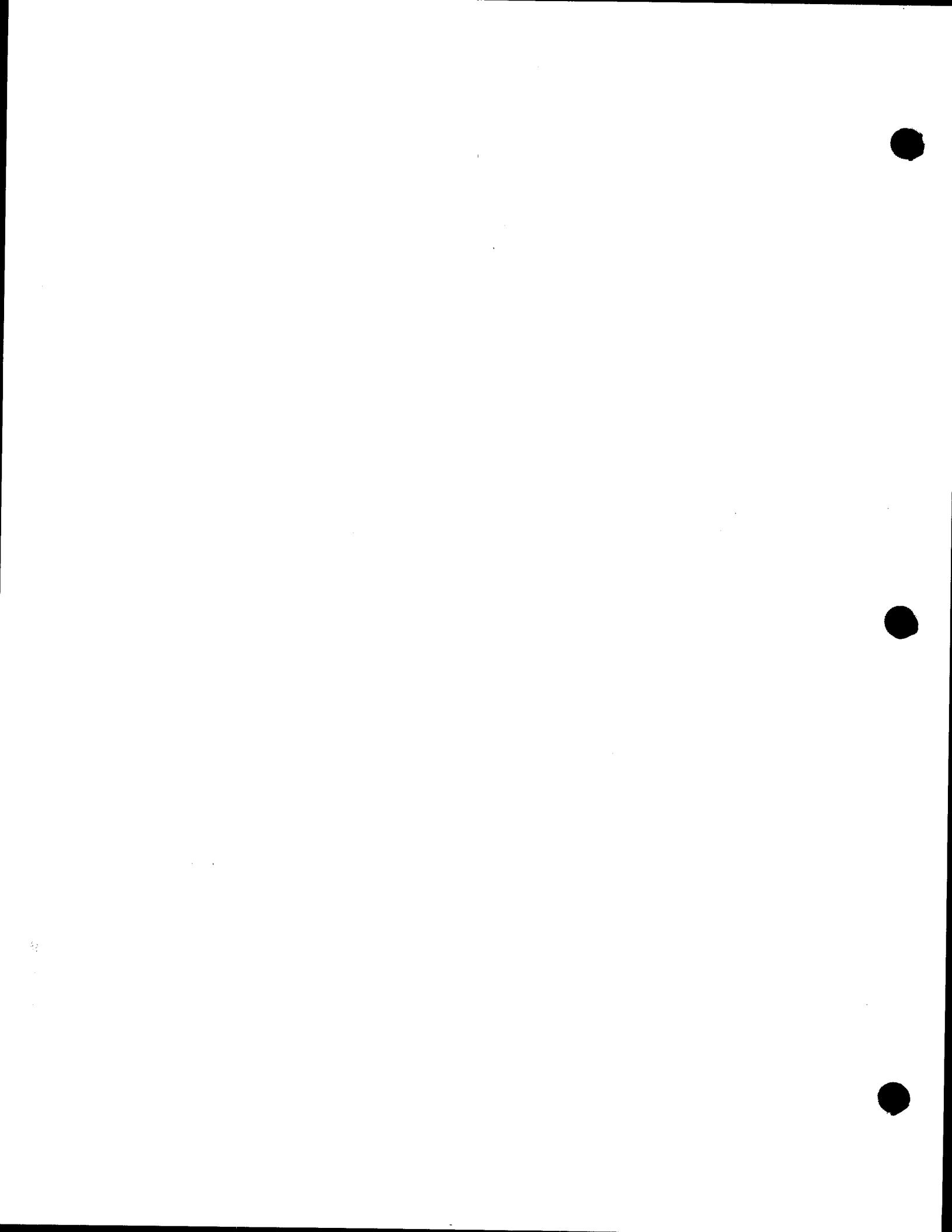
Indicates end of input data.

- Function key 16

Indicates input data cannot be entered properly.

- Function key 20

Restores the screen to its original contents for the current pass if it has inadvertently been destroyed.



Appendix D. Alphabet Table

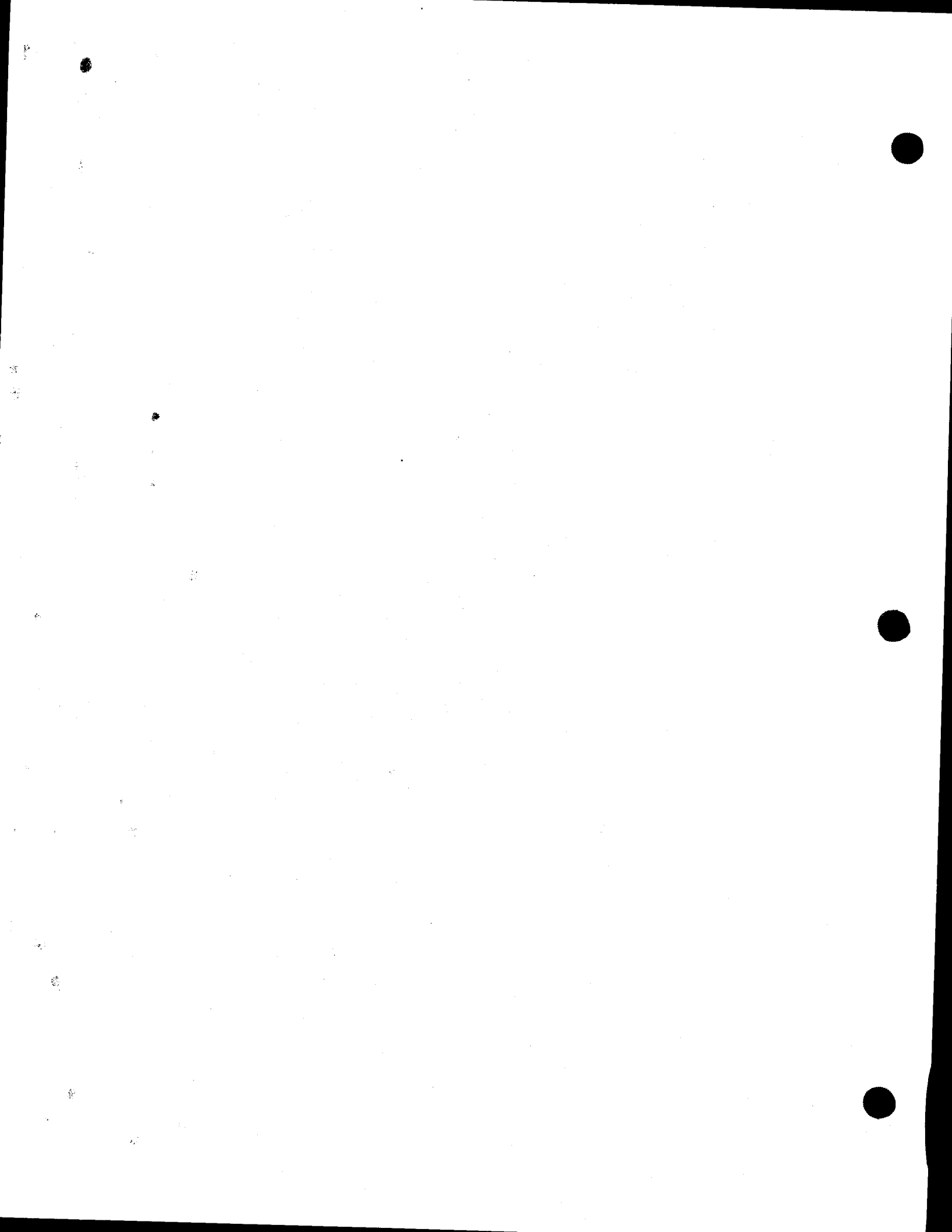
The following table indicates characters considered alphabetic in addition to A-Z, a-z, and the space character.

| Keyboard | Character | ASCII | EBCDIC |
|--------------------|-----------|-------|--------|
| Denmark and Norway | Å | 5D | 4F |
| | å | 7D | D0 |
| | Ø | 5C | E0 |
| | ø | 7C | 6A |
| | Æ | 5B | 4A |
| | æ | 7B | C0 |
| France | ç | 5C | E0 |
| | é | 7B | C0 |
| | è | 7D | D0 |
| | ù | 7C | 6A |
| | à | 40 | 7C |
| Germany | Ä | 5B | 4A |
| | Ö | 5C | E0 |
| | Ü | 5D | 4F |
| | ä | 7B | C0 |
| | ö | 7C | 6A |
| | ü | 7D | D0 |
| | ß | 7E | A1 |
| Italy | à | 7B | C0 |
| | è | 7D | D0 |
| | ì | 7E | A1 |
| | ò | 7C | 6A |
| | ù | 60 | 79 |
| | é | 5D | 4F |
| Sweden and Finland | Ä | 5B | 4A |
| | Ö | 5C | E0 |
| | Å | 5D | 4F |
| | É | 40 | 7C |
| | Ü | 5E | 5F |
| | ä | 7B | C0 |
| | ö | 7C | 6A |
| | å | 7D | D0 |
| | é | 60 | 79 |
| | ü | 7E | A1 |
| Spain | Ñ | 5C | E0 |
| | ñ | 7C | 6A |



Appendix E. SFS Support of Display Attributes

| Display Attribute | Device | | | | |
|----------------------|------------------|------------------|-----------------------------------|-------------------|---|
| | U 200 | U 400 | UTS/20D UTS/20 | UTS/40D UTS/40 | Comments |
| Display Intensity: | | | | | |
| Normal | Normal intensity | Normal intensity | Normal intensity | Normal intensity | |
| Alternate Brightness | Normal intensity | Low intensity | Low/reverse/normal (See comment.) | Low intensity | Dependent upon control page setting |
| Low | Normal intensity | Low intensity | Low/reverse/normal (See comment.) | Low intensity | Dependent upon control page setting |
| Nondisplay | Normal intensity | Nondisplay | Normal intensity | Nondisplay | |
| Highlighting: | | | | | |
| Blink | Blink character | Blink field | Blink field (See comment.) | Blink field | Blink character for Katakana UTS/20D & UTS/20 |
| Reverse video | Blink character | Blink field | Blink field | Reverse video | Blink character for Katakana UTS/20D & UTS/20 |
| Emphasis: | | | | | |
| Underline | N/A | N/A | N/A | Underline | |
| Strike thru | N/A | N/A | N/A | Strike thru | |
| Column separator | N/A | N/A | N/A | Column separator | |



Appendix F. Sample IMS Action COBOL Program for Screen Format Services (JAMENU)

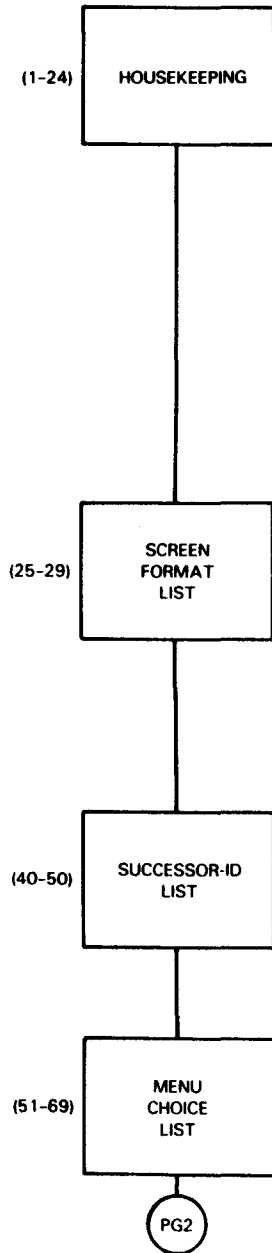
The JAMENU action program processes a password entered as input from the terminal. If the password is valid, JAMENU displays a menu screen using screen format services.

The operator then chooses the menu number of the action program he needs to perform the next operation on his file. If the password he enters is invalid, JAMENU displays an error screen and terminates.

Figure F-1 is a compiler listing of the JAMENU action program. Because this program is one in a series of interrelated action programs, note that a special function call section (lines 269-363) includes many more calls than JAMENU uses. Including a repertoire of these calls in each action program makes them available for any logic used in each procedure division of programs in the series.

Also, in the working-storage section, all screen formats and successor-ids are identified enabling the program to reference any one of them, though it does not use all of them. This programming technique saves time particularly when a series of action programs can succeed differently to each other.

A flowchart corresponding to the JAMENU action program appears to the left of the coding in Figure F-1. Program line numbers in parentheses near the flowchart boxes represent the lines of coding that implement the process described.



```

LINE NO. SOURCE ENTRY
000001 IDENTIFICATION DIVISION.
000002
000003 PROGRAM-ID. JAMENU.
000004 REMARKS. PROCESS SIGNON + MENU.
000005 -----
000006 THIS PROGRAM PROCESSES THE SIGNON AND SYSTEM/80 MENU *
000007 SCREEN FOR THE ENTITLEMENT ACCOUNTING SYSTEM. *
000008 IF THE SIGNON IS FOUND TO BE VALID, THE MENU WILL BE *
000009 DISPLAYED. OTHERWISE, THE ERROR OVERLAY SCREEN WILL BE *
000010 DISPLAYED AND THE TRANSACTION TERMINATED. *
000011 -----
000012
000013 ENVIRONMENT DIVISION.
000014
000015 CONFIGURATION SECTION.
000016 SOURCE-COMPUTER. UNIVAC-053.
000017 OBJECT-COMPUTER. UNIVAC-053.
000018
000019 DATA DIVISION.
000020
000021 WORKING-STORAGE SECTION.
000022 77 CUST-FILENAME PIC X(7) VALUE 'CUSTMST'.
000023 77 SCTL-FILENAME PIC X(7) VALUE 'SYSCTL'.
000024
000025 01 SCREEN-FORMAT-IDS.
000026 05 SF-MENU PIC X(8) VALUE 'JASMFNU'.
000027 05 SF-ADD1 PIC X(8) VALUE 'JASADD1'.
000028 05 SF-ADD2 PIC X(8) VALUE 'JASADD2'.
000029 05 SF-ADD3 PIC X(8) VALUE 'JASADD3'.
000030 05 SF-CHG1 PIC X(8) VALUE 'JASCHG1'.
000031 05 SF-CHG2 PIC X(8) VALUE 'JASCHG2'.
000032 05 SF-CHG3 PIC X(8) VALUE 'JASCHG3'.
000033 05 SF-DEL1 PIC X(8) VALUE 'JASDEL1'.
000034 05 SF-DIS1 PIC X(8) VALUE 'JASDIS1'.
000035 05 SF-LST1 PIC X(8) VALUE 'JASLST1'.
000036 05 SF-WAR1 PIC X(8) VALUE 'JASWAR1'.
000037 05 SF-ERR1 PIC X(8) VALUE 'JASERR1'.
000038 05 SF-TERM PIC X(8) VALUE 'JASTERM'.
000039
000040 01 VALID-SUCCESSOR-IDS.
000041 05 MENU PIC X(6) VALUE 'JAMFNU'.
000042 05 CUST-ADD PIC X(6) VALUE 'JAADD1'.
000043 05 CUST-CHG-1 PIC X(6) VALUE 'JACHG1'.
000044 05 CUST-CHG-2 PIC X(6) VALUE 'JACHG2'.
000045 05 CUST-CHG-3 PIC X(6) VALUE 'JACHG3'.
000046 05 CUST-DEL PIC X(6) VALUE 'JADFL1'.
000047 05 CUST-DISPLAY PIC X(6) VALUE 'JADIS1'.
000048 05 CUST-LIST PIC X(6) VALUE 'JALST1'.
000049 05 WS-ACTIVITY PIC X(6) VALUE 'JAWARI'.
000050
000051 01 WS-TABLES.
000052 05 MENU-TABLE.
000053 10 FILLER PIC X(9) VALUE '01JAADD1'.
000054 10 FILLER PIC X(9) VALUE '02JACHG1'.
000055 10 FILLER PIC X(9) VALUE '03JACHG2'.
000056 10 FILLER PIC X(9) VALUE '04JACHG3'.
000057 10 FILLER PIC X(9) VALUE '05JADFL1'.
000058 10 FILLER PIC X(9) VALUE '06JADIS1'.
000059 10 FILLER PIC X(9) VALUE '07JALST1'.
000060 10 FILLER PIC X(9) VALUE '08JAWAR1'.
000061 10 FILLER PIC X(9) VALUE '09JAMENUN'.
000062 10 FILLER PIC X(9) VALUE '10JAMENUT'.
000063
000064 05 MENU-TFL REDEFINES MENU-TABLE OCCURS 10 TIMES
000065 INDEXED BY MENU-INDX.
000066 10 MENU-SEL PIC 9(2).
000067 10 MENU-NAME PIC X(6).
000068 10 MENU-IND PIC X.
000069
000070 *****
000071 THIS IS THE ERP PROCESSING TABLE FOR RETRIEVING ERP
000072 MESSAGES FROM THE SYSTEM CONTROL FILE FOR DISPLAY
000073 WITH THE OVERLAY.
000074 *****
  
```

Figure F-1. Sample Action Program JAMENU Using Screen Formats (Part 1 of 6)

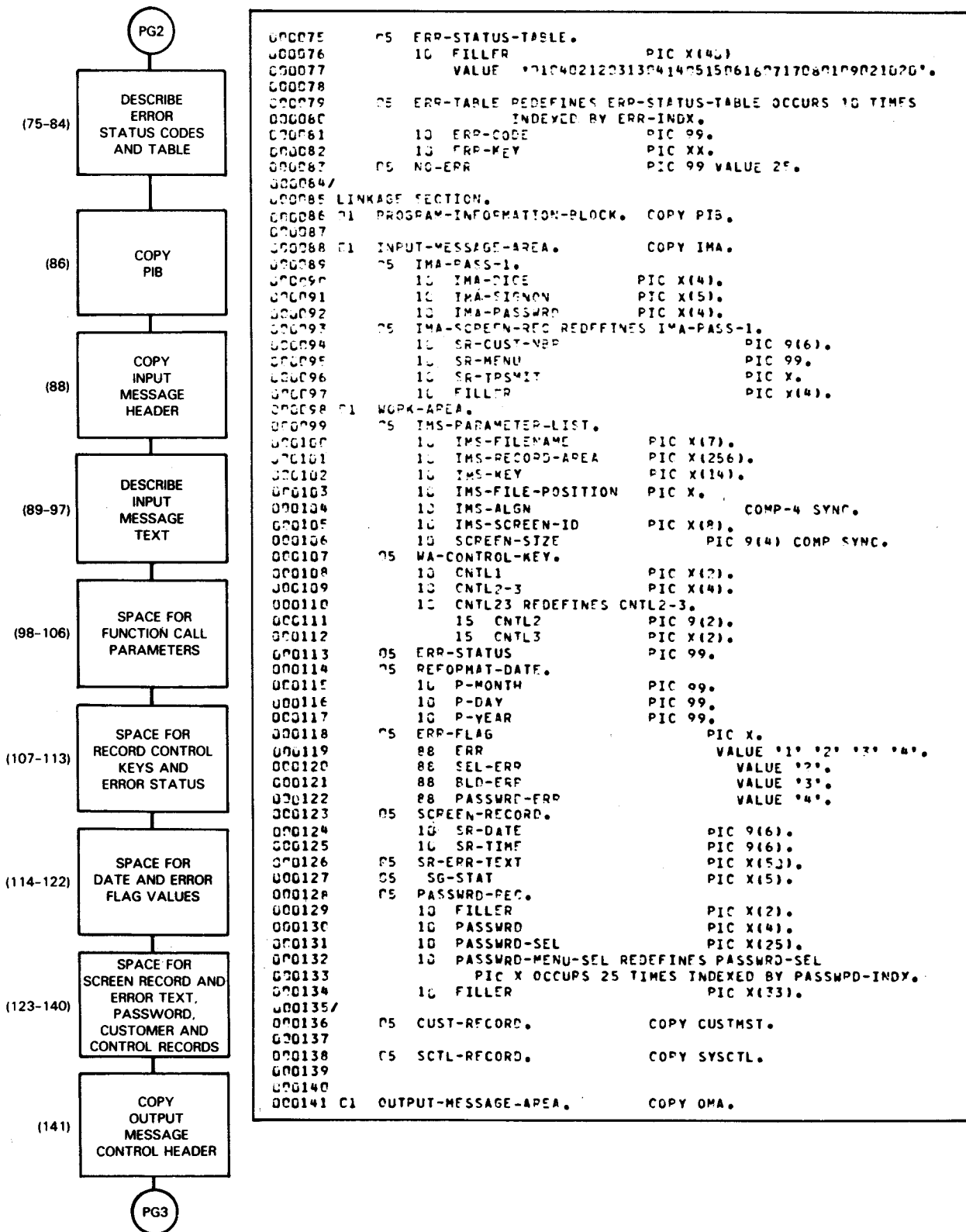
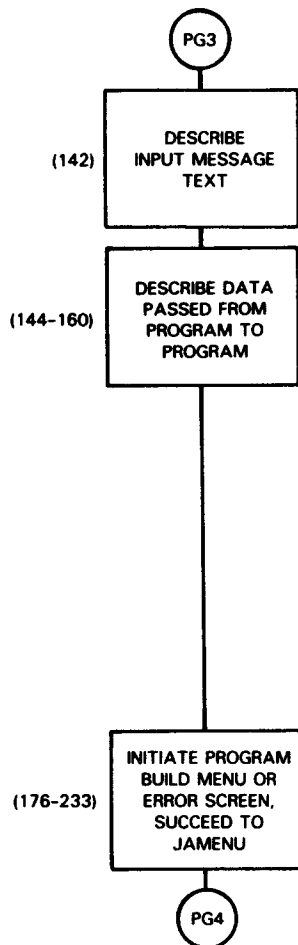


Figure F-1. Sample Action Program JAMENU Using Screen Formats (Part 2 of 6)



```

000142 05 OMA-TEXT PIC X(3000).
000143
000144 01 CONTINUITY-DATA-AREA.
000145 05 CDA-PASSWRD PIC X(4).
000146 05 CDA-MENU-SFL PIC X(25).
000147 05 CDA-PASSWRD-MENU-SEL REDEFINES CDA-MENU-SFL
000148 PIC X OCCURS 25 TIMES.
000149 05 CDA-CUST-KEY.
000150 10 CDA-CUST-NPR PIC 9(6).
000151 05 CDA-ACCT-CODF PIC X(4).
000152 05 PASS-FLAG PIC X.
000153 88 PASS-THRU VALUE '1''2''3''4''5'.
000154 88 PASS1 VALUE '1'.
000155 88 PASS2 VALUE '2'.
000156 88 PASS3 VALUE '3'.
000157 88 PASS4 VALUE '4'.
000158 88 PASS5 VALUE '5'.
000159 05 CDA-STATUS-BYTE PIC X.
000160 05 CDA-PROGRAM-NAME PIC X(16).
000161
000162 PROCEDURE DIVISION USING PROGRAM-INFORMATION-POOL
000163 INPUT-MESSAGE-AREA
000164 WORK-AREA
000165 OUTPUT-MESSAGE-AREA
000166 CONTINUITY-DATA-AREA.
000167 MAIN-LOOP SECTION.
000168 *****
000169* THE PASS FLAG IN THIS SECTION TELLS THE PROGRAM AT WHICH
000170* POINT IMS HAS RETURNED CONTROL OF THE PROCESSING TO THE
000171* PROGRAM A PASS 2 FLAG MEANS THE PROGRAM HAS ALREADY PUT
000172* OUT THE SCREEN AND IS NOW READY TO ACCEPT THE DATA FROM
000173* THAT SCREEN TO PROCESS. OTHERWISE THE PROGRAM WANTS TO
000174* DO THE INITIAL PROCESSING TO PUT OUT A SCREEN.
000175* *****
000176 DO-BEGIN.
000177 IF NOT PASS?
000178 PERFORM 100-INITIALIZE
000179 IF NOT ERR
000180 PERFORM 200-BUILD-SCREEN
000181 ELSE
000182 PERFORM 900-ERR-MESSAGE
000183 ELSE
000184 PERFORM 250-READ-SCREEN.
000185 PERFORM 500-RETURN.
000186
000187 *****
000188* INITIALIZATION OF FIELDS AND FLAGS IS DONE HERE.
000189* ALSO CHECKING IS DONE TO SEE IF THE PROGRAM ENTERED
000190* FROM A SIGN ON OR WAS CALLED FROM ANOTHER PROGRAM.
000191* ONLY IF ENTER FROM A SIGN ON DOES THE PROGRAM RETRIEVE
000192* THE PASSWORD RECORD. OTHERWISE IT IS CARRIED TO CHECK
000193* VALIDITY OF MENU SELECTION.
000194* *****
000195 100-INITIALIZE.
000196 MOVE SPACE TO IMS-KEY
000197 IMS-FILENAME
000198 SR-ERR-TEXT.
000199
000200 MOVE '2' TO PASS-FLAG.
000201 MOVE '0' TO ERR-FLAG.
000202
000203 MOVE CORP P-TRANSACTION-DATE TO REFORMAT-DATE.
000204 IF CDA-PROGRAM-NAME EQUAL LOW-VALUES
000205 MOVE IMA-PASSWPD TO CNTL2-3
000206 MOVE 'PW' TO CNTL1
000207
000208 MOVE SPACE TO IMS-RECORD-AREA
000209 MOVE SCTL-FILENAME TO IMS-FILENAME
000210 MOVE WA-CONTROL-KEY TO IMS-KEY
000211 PERFORM 502-GET
000212 IF ERR
000213 MOVE '4' TO ERR-FLAG
000214 ELSE
000215 MOVE IMS-RECORD-AREA TO PASSWRD-REC
000216 MOVE PASSWRD-SEL TO CDA-MENU-SEL
000217 MOVE PASSWRD TO CDA-PASSWPD.
000218
000219 MOVE MFNU TO CDA-PROGRAM-NAME.
  
```

Figure F-1. Sample Action Program JAMENU Using Screen Formats (Part 3 of 6)

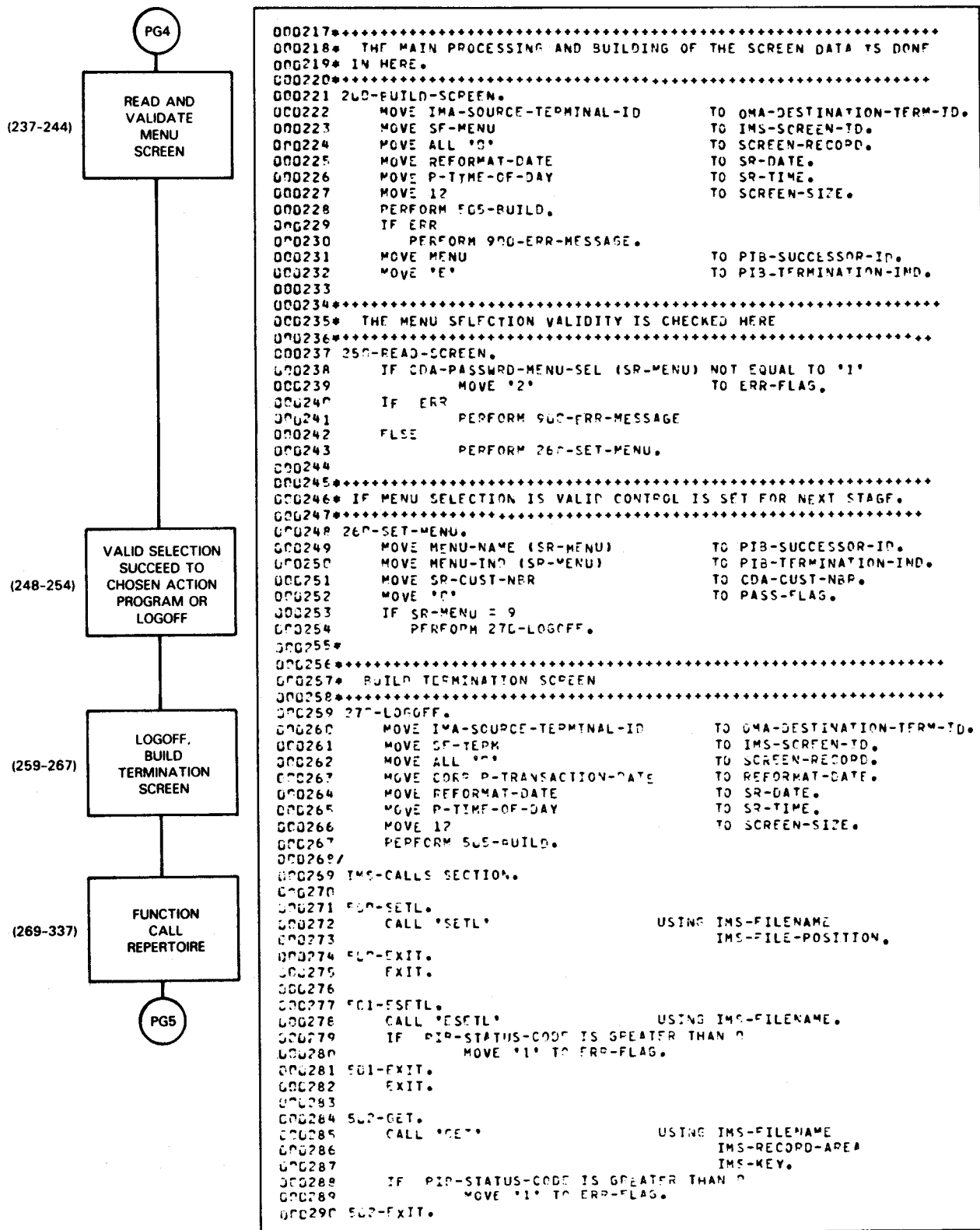


Figure F-1. Sample Action Program JAMENU Using Screen Formats (Part 4 of 6)

PG5

```

J00291 EXIT.
J00292
J00293 503-SETUP.
J00294 CALL 'GETLP' USING IMS-FILENAME
J00295 IMS-RECORD-AREA
J00296 IMS-KEY.
J00297 IF PIB-STATUS-CODE IS GREATER THAN 0
J00298 MOVE '1' TO ERR-FLAG.
J00299 503-EXIT.
J00300 EXIT.
J00301
J00302 504-PUT.
J00303 CALL 'PUT' USING IMS-FILENAME
J00304 IMS-RECORD-AREA.
J00305 IF PIB-STATUS-CODE IS GREATER THAN 0
J00306 MOVE '1' TO ERR-FLAG.
J00307 504-EXIT.
J00308 EXIT.
J00309
J00310 *****
J00311 * CALL FOR MAIN SCREEN FOR PROGRAM
J00312 *****
J00313 505-BUILD.
J00314 CALL 'BUILD' USING OUTPUT-MESSAGE-AREA
J00315 IMS-SCREEN-ID
J00316 SCREEN-RECORD
J00317 SCREEN-SIZE
J00318 SG-STAT.
J00319 IF PIB-STATUS-CODE IS GREATER THAN 0
J00320 MOVE '3' TO ERR-FLAG.
J00321 505-EXIT.
J00322 EXIT.
J00323
J00324 *****
J00325 * CALL FOR EPR OVERLAY SCREEN
J00326 *****
J00327 505-BUILD-ERR.
J00328 CALL 'BUILD' USING OUTPUT-MESSAGE-AREA
J00329 IMS-SCREEN-ID
J00330 SR-ERR-TEXT
J00331 SCREEN-SIZE
J00332 SG-STAT.
J00333 IF PIB-STATUS-CODE IS GREATER THAN 0
J00334 MOVE '3' TO ERR-FLAG.
J00335 505-BLD-ERR-EXIT.
J00336 EXIT.
J00337
J00338 506-REBUILD.
J00339 CALL 'REBUILD' USING IMS-SCREEN-ID
J00340 IMS-RECORD-AREA.
J00341 506-EXIT.
J00342 EXIT.
J00343
J00344 507-RETURN.
J00345 CALL 'RETURN'.
J00346 507-EXIT.
J00347 EXIT.
J00348
J00349 508-INSEPT.
J00350 CALL 'INSEPT' USING IMS-FILENAME
J00351 IMS-RECORD-AREA
J00352 IMS-KEY.
J00353 IF PIB-STATUS-CODE IS GREATER THAN 0
J00354 MOVE '1' TO ERR-FLAG.
J00355 508-EXIT.
J00356 EXIT.
J00357
J00358 599-SNAP.

```

FUNCTION
CALL
REPERTOIRE

(338-362)

PG6

Figure F-1. Sample Action Program JAMENU Using Screen Formats (Part 5 of 6)

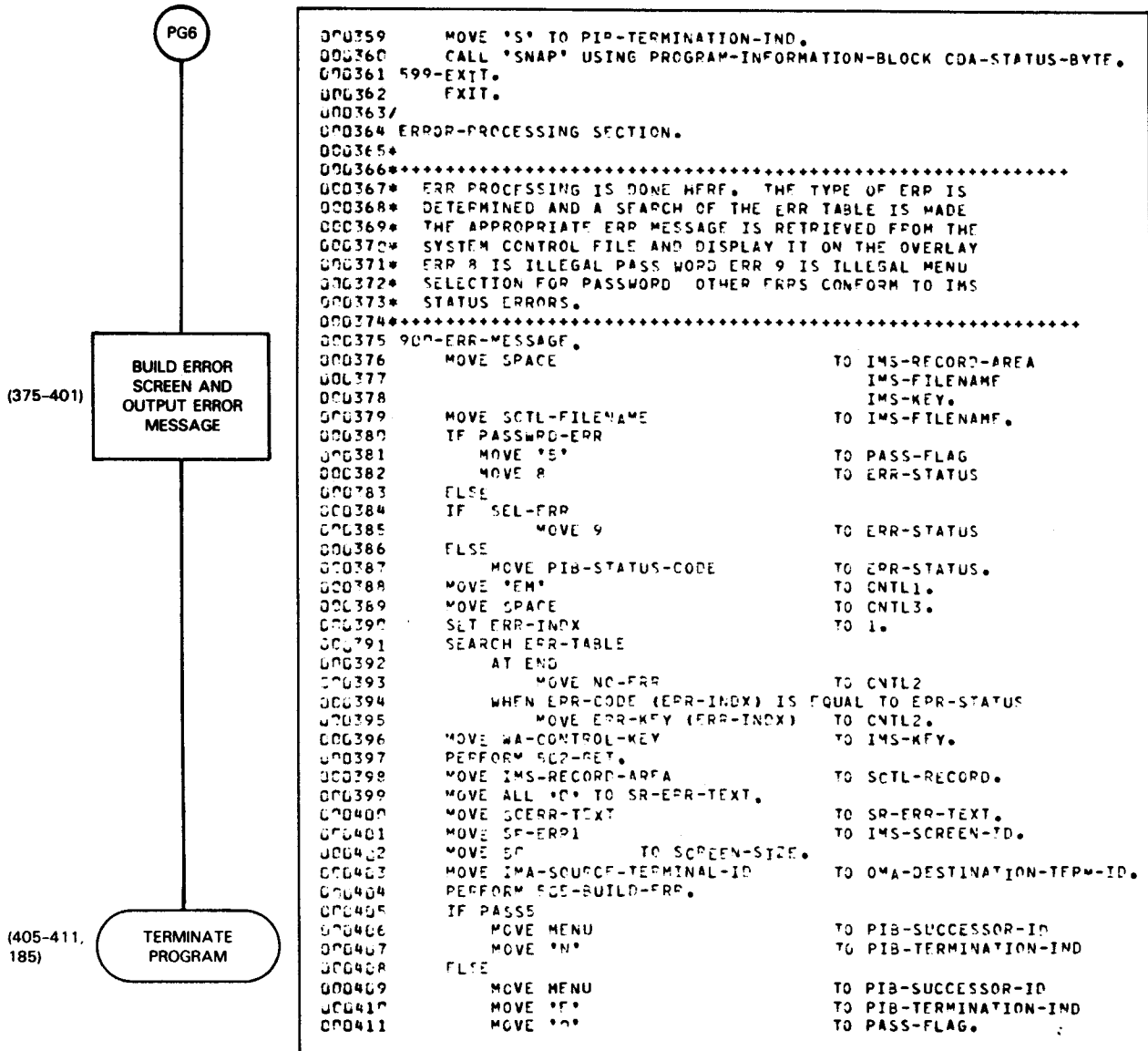


Figure F-1. Sample Action Program JAMENU Using Screen Formats (Part 6 of 6)

The following discussion of the JAMENU action program assumes that you have already created a menu screen format called JA\$MENU and filed it in the screen format file. Any line numbers referenced in this discussion refer to the code in the JAMENU action program, Figure F-1. Also, expansions of the program information block, input message area, and output message area cannot be seen in this listing; however, their fields may be referenced in the code (e.g., lines 406 and 407) and are available to JAMENU.

JAMENU uses two files (lines 22 and 23):

1. CUSTMST file
2. SYSCTL file

The CUSTMST file contains customer information. The SYSCTL file contains four types of records:

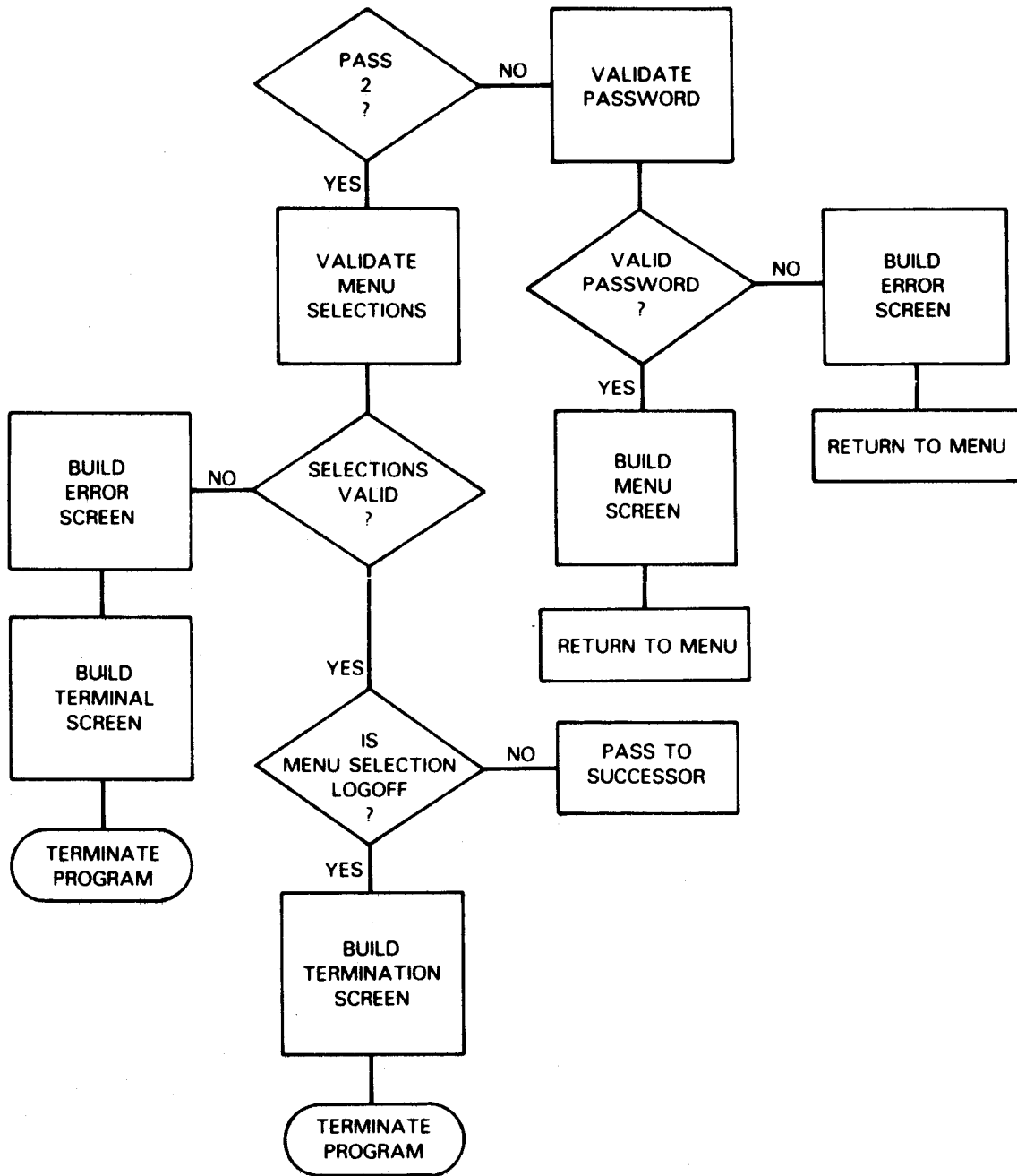
1. Account access records (AA)
2. Branch records (BR)
3. Error message text records (EM)
4. Password records (PW)

Each type record is identified by a 2-byte control key field. (See lines 108-112 and 129.) JAMENU accesses the SYSCTL file to validate passwords and retrieve error messages for display in the error message screen format.

JAMENU performs five types of routines. It:

1. validates passwords;
2. builds menu screen;
3. validates menu selections;
4. builds error screen; and
5. builds termination screen

The following general flowchart shows these main routines in the JAMENU program.



JAMENU GENERAL FLOWCHART

Begin executing the JAMENU program by entering the transaction code, MENU, followed by the password. This is considered the sign-on or first pass through JAMENU.

MENU CP50

On the first pass, JAMENU accesses the SYSCTL file to validate the password entered at the terminal. If the password is valid, JAMENU saves all data pertinent to that password in the continuity data area (line 211-216), builds the menu screen (lines 221-232), and terminates in external succession to itself (JAMENU). Menu screen JA\$MENU follows.

FIRST PASS

```

06/23/81      06:49:28      JAMENU      02/09/81
                ENTITLEMENT ACCOUNTING SYSTEM
SELECT ONE (1) OF THE FOLLOWING OPTIONS:
  1.  ADD A NEW CUSTOMER RECORD.
*2.  UPDATE CUSTOMER NAME/ADDRESS INFORMATION.
*3.  UPDATE BRANCH CUSTOMER INFORMATION.
*4.  UPDATE CUSTOMER ENTITLEMENTS.
*5.  DELETE A CUSTOMER RECORD.
*6.  DISPLAY CUSTOMER INFORMATION.
  7.  LIST ALL ACCOUNTS (ON THE WORKSTATION).
  8.  ENTER WORKSTATION ACTIVITY RECORDS.
  9.  LOGOFF SYSTEM.
*ENTER CUSTOMER NUMBER  -----
MENU SELECTION:  --
PLACE CURSOR HERE TO TRANSMIT  [-]

```

In the menu screen build routine (lines 221-232), the BUILD function call that actually calls the menu screen identifies the buffer address where IMS receives the screen format as the output message area (line 314); the format name as IMS-SCREEN-ID (line 315, defined on line 105); the variable data as SCREEN-RECORD (line 316, defined on lines 123-125); the data size as SCREEN-SIZE (line 317, defined on line 106); and, the output status as SG-STAT (line 318, defined on line 127).

Notice, all the parameters you specify on the BUILD function must be defined in the work area.

If the BUILD function is unsuccessful (lines 319 and 320), JAMENU moves an error code of 3 to the ERR-FLAG (lines 118 and 121) indicating a build error.

If the password is invalid on the first pass, JAMENU accesses the SYSCTL file via the EM record key for the error message record (lines 380-388), searches an error table to find the appropriate error message (lines 390-395), retrieves that error message (lines 396-398), builds the error message screen (lines 399-404), and terminates in external succession to itself (lines 408-411). The password error screen follows:

```
PASSWORD IS INVALID. ENTER AGAIN.
```

On the second pass through JAMENU, the program tests the menu selection made, to see if it is accessible to the password specified in the first pass. If the menu selection is valid for that password, JAMENU performs 260-SET-MENU (lines 248-255). This moves the correct program name to process the menu selection to the successor-id and an I to the termination indicator.

Notice here that the programmer has set up a menu table (lines 52-62) containing not only the menu selection numbers and their corresponding action programs but also the termination indicators used to end each action program. The menu is redefined with selection numbers (MENU-SEL) in the first two bytes of each table field, the action program names are in the next 6 bytes (MENU-NAME), and, finally, the termination indicators are in the last byte of each field (MENU-IND).

When the program moves the successor-id and termination indicator to the program information block (lines 248-250), it moves the menu name indexed by the menu number entered at the terminal. JAMENU picks up the correct program name for the successor-id by using this index value to reference the first two bytes of the menu table entry. Likewise, JAMENU moves the termination indicator value to the program information block by using the index value to reference the last byte of the menu table entry chosen.

Redefining the menu table (lines 52-68) saves coding by making three types of data accessible in one table: the menu selection numbers, action program names for successor-ids, and termination indicators.

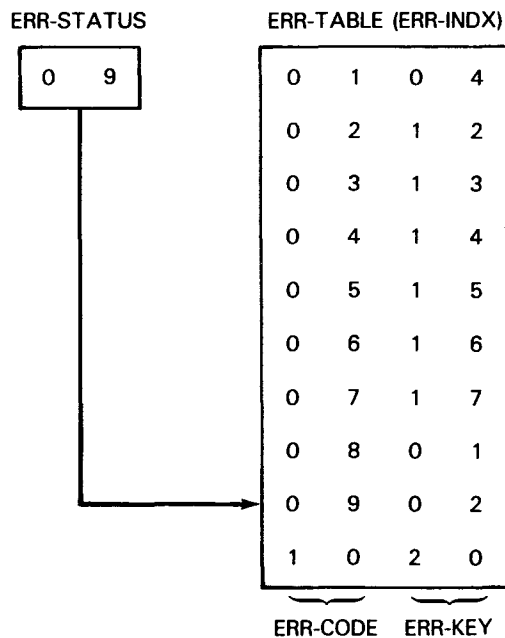
If the menu selection is invalid, JAMENU moves code 2 indicating selection error to ERR-FLAG (lines 237-241), builds the menu selection error message screen (lines 375-411), and succeeds externally to itself.

Several tests occur in the beginning error message building routine. The first separates password errors from menu selection errors and function call errors (lines 380-387).

For a password error, JAMENU places code 5 in the pass flag to force the normal termination of the transaction and moves 8 to the work area location, ERR-STATUS (lines 380-382).

For a menu selection error, JAMENU moves a 9 to ERR-STATUS in the work area (lines 113, 384, and 385). This code corresponds to one of the values 01 through 10 contained in the first two bytes of each table entry in the ERR-TABLE. These leading two bytes in each table entry also correspond to the index value being used to search ERR-TABLE (lines 75-83). Thus, when the value in ERR-STATUS equals the value in the first two bytes of an ERR-TABLE entry, JAMENU moves the contents of ERR-KEY (the last two bytes in the corresponding ERR-TABLE entry) to the record key area used to retrieve that error message record from the SYSCLT file (lines 394 and 395).

The following diagram illustrates the ERR-TABLE, its index (ERR-INDX), and the way JAMENU uses the value in ERR-STATUS to find the ERR-KEY value in the table by searching ERR-TABLE for the error code (ERR-CODE) that matches the value in ERR-STATUS.



JAMENU clears the work-area locations (lines 376-378). It moves the SYSCTL file name to the work area file name to prepare for retrieval of the SYSCTL record. This record contains the 'EM' prefix, the error message number to be sent to the screen, and the error message text (line 379).

To find the appropriate error message corresponding to the password error menu selection error, or other function call error, JAMENU searches the table, ERR-TABLE (lines 390-395). If it finds no corresponding error code, it moves a message number of 25 (line 83) to the key field (CNTL-2, line 395) used to call the corresponding record from the SYSCTL error message file (lines 396 and 397 and 284-289).

If, for example, JAMENU finds an 09 error code (lines 394 and 395), JAMENU uses error message number 02 from the ERR-TABLE (see ERR-TABLE diagram and coding line 77) as a key to locate the corresponding error message text in the SYSCTL file (lines 102, 107-112, and 396 and 397).

When JAMENU retrieves the SYSCTL error message (EM) record, it uses this message number to locate the error message text immediately following the 02 error number on the SYSCTL record. JAMENU then uses this message text in building the error message screen.

Notice in lines 398-404, including lines 327-334, that JAMENU clears the screen error text area to receive the error message text from the SYSCTL file; identifies the terminal to receive the error message; transmits the message; and terminates in external succession to itself. If a build error occurs, JAMENU sets the error flag to 3 and succeeds externally to itself.

If the menu selection including customer number is valid, JAMENU executes another short routine (260-SET-MENU, lines 248-254) that passes control to the appropriate action program to process the menu selection. This routine also checks for a logoff menu selection (9) that builds the termination screen similarly to the way JAMENU built the error message screen (lines 259-267). Successor programs selected from the menu perform file operations required. When processing is complete, control returns to the JAMENU program via immediate internal succession and the terminal operator again receives the menu screen to enter another selection.



Appendix G. Batch Format Generator

Instead of using interactive SFGEN, you can use the Batch Format Generator to spool one, some, or all of the screen formats in a particular library. Following is the format for the Batch Format Generator. Use keyword parameters where necessary.

```
{RUN} BFGSPL, [ ,SAVE={YES} ] [ ,FI={input file name} ]
{RV}      [ ,SAVE={NO} ] [ ,FI={SYSTEM} ]

[ ,VI={vsn of input file} ] [ ,FMT={xxxxxxx} ]
[ ,VI={RES} ] [ ,FMT={xxxxxxx.} ]
```

where:

SAVE=YES

Saves the RUN environment exactly as built for this execution by job control. At a later time, this environment may be activated by the SCHEDULE (SC) command.

NOTE:

The advantage of SC over RUN (RV) is that less processing time is required.

SAVE=NO

RUN environment is not saved. This is the default.

FI=input file name

Names the user input file, which must be a MIRAM file.

FI=SYSTEM

If no FI= is specified, defaults to FI=\$Y\$FMT file.

VI=vsn of input file

Names the user input file vsn, which must be a disk.

**VI=RES**

If no VI= is specified, defaults to the VI=RES (boot) file.

FMT=xxxxxxx

Spools only a specific screen format where xxxxxxxx is the format name.

FMT=xxxxxxx.

Spools screen formats by prefix where xxxxxxx. is the desired prefix.

FMT=

Spools all formats in a library. This is the default.



| Term | Reference | Page |
|---|-----------|------|
| Bidirectional fields | | |
| characteristics screen | 3.3.10 | 3-11 |
| conditional display | 3.5.3 | 3-38 |
| conditional protection | 3.5.6 | 3-44 |
| conditional retention | 3.5.5 | 3-43 |
| dialog screen 3 | 3.5.3 | 3-38 |
| dialog screen 4 | 3.5.4 | 3-41 |
| dialog screen 5 | 3.5.5 | 3-43 |
| dialog screen 6 | 3.5.6 | 3-44 |
| display retention | 3.3.10 | 3-11 |
| display retention screen | 3.4.6 | 3-24 |
| ESCORT | 7.3.5 | 7-35 |
| IMS | 7.3.6 | 7-36 |
| list screen | 6.1.2.6 | 6-9 |
| nondisplay screen | 3.4.8 | 3-27 |
| RPG II | 7.3.1 | 7-5 |
| | 7.3.6 | 7-36 |
| SHOW function | 6.1.1 | 6-1 |
| specifying | 1.3.2.2 | 1-10 |
| | 4.1.3 | 4-10 |
| See also I/O fields and variable data fields. | | |
| Blank character | | |
| edit screen | 3.4.3 | 3-19 |
| editing alphabetic and alphanumeric fields | 2.3.1 | 2-2 |
| simple insertion in numeric fields | 2.3.2 | 2-3 |

| Term | Reference | Page |
|---|-----------|------|
| C | | |
| CD\$DDINF + 11 error codes | | |
| DMINP imperative macroinstruction | 7.3.4.3 | 7-31 |
| DMOUT imperative macroinstruction | 7.3.4.5 | 7-35 |
| DMSEL,SCREEN imperative macroinstruction | 7.3.4.4 | 7-33 |
| CHANGE I/O option | | |
| modification summary | 5.2 | 5-6 |
| modify screen | 5.1.1 | 5-2 |
| | 5.1.1.3 | 5-4 |
| CHANGE TEMPLATE ONLY - NO INTERNAL CHANGES option | | |
| modification summary | 5.2 | 5-6 |
| modify screen | 5.1.1 | 5-2 |
| | 5.1.1.4 | 5-4 |
| CHANGE TEMPLATE option | | |
| modification summary | 5.2 | 5-6 |
| modify screen | 5.1.1 | 5-2 |
| | 5.1.1.1 | 5-3 |
| CHANGE TYPE option | | |
| modification summary | 5.2 | 5-6 |
| modify screen | 5.1.1 | 5-2 |
| | 5.1.1.2 | 5-4 |
| Characteristics screen | | |
| alphabet (line 4) | 3.3.2 | 3-7 |
| creating new format | 3.3 | 3-6 |
| | 4.1 | 4.2 |
| cursor positioning and overwriting | 3.6 | 3-49 |
| display retention (line 20) | 3.3.10 | 3-11 |
| erasing and unlocking options (lines 10, 11, and 12) | 3.3.5 | 3-8 |
| error message field (line 19) | 3.3.9 | 3-11 |
| error retry counts (line 3) | 3.3.1 | 3-7 |
| function keys (line 21) | 3.3.11 | 3-11 |
| lowercase translation (line 6) | 3.3.3 | 3-7 |
| nondisplayed field (line 11) | 3.3.11 | 3-11 |
| original and overlay formats (line 8) | 3.3.4 | 3-8 |
| prompting additional screens | 3.3 | 3-6 |
| | 3.4 | 3-12 |
| related optional and dialog screens | Table 3-1 | 3-12 |
| | 3.4 | 3-12 |
| | 3.5 | 3-28 |
| sample format | 4.2.1.1 | 4-15 |
| special display control (line 17) | 3.3.8 | 3-10 |
| special editing characters (line 16) | 3.3.7 | 3-10 |
| CLOSE (close a file), BAL imperative macroinstruction | 7.3.4.2 | 7-29 |

| Term | Reference | Page | Term | Reference | Page |
|--|---|------|---|-----------|------|
| COBOL | | | Conditional retention dialog (dialog screen 5) | 3.5.5 | 3-43 |
| connect-free reporting | Table 7-2 | 7-24 | Conditional values screen | 6.1.2 | 6-2 |
| data division | 7.3.2.1 | 7-10 | | 6.1.2.8 | 6-12 |
| environment division | 7.3.2.1 | 7-10 | CR character | | |
| function key reporting | Table 7-3 | 7-25 | dialog screen 1 | 3.5.1 | 3-31 |
| function keys and multivolume workstations | 7.3.2.3 | 7-21 | editing rules for signed output fields | Table 2-1 | 2-6 |
| indicators | 7.3.2.2 | 7-18 | fixed insertion in numeric fields | 2.3.2 | 2-3 |
| interprogram communication considerations | 7.3.2.4 | 7-26 | CREATE-FROM function | | |
| major coding considerations | 7.3.2.1 | 7-10 | characteristics screen | 3.3.2 | 3-7 |
| procedure division | 7.3.2.1 | 7-11 | dialog screen 1 | 3.5.1 | 3-31 |
| status keys | Table 7-1 | 7-22 | home screen | 3.2 | 3-2 |
| Comma | | | | 3.2.1 | 3-3 |
| dialog screen 1 | 3.5.1 | 3-34 | modifying formats | 5.1 | 5-1 |
| editing input or bidirectional fields | 1.3.2.3 | 1-11 | | 5.2 | 5-6 |
| protecting | 3.5.1 | 3-35 | new format name | 3.2.2 | 3-4 |
| simple and fixed insertion in numeric fields | 2.3.2 | 2-3 | old format name and library | 3.2.3 | 3-5 |
| Command keys | See function/ command key screen. | | CREATE function | | |
| Conditional display dialog (dialog screen 3) | 3.5.3 | 3-38 | basic screen format | 4.1 | 4-1 |
| Conditional display properties dialog (dialog screen 4) | 3.5.4 | 3-40 | characteristics screen | 3.3 | 3-6 |
| Conditional erase/replenish screen | | | | 4.1 | 4-2 |
| characteristics screen | Table 3-1 | 3-12 | complex screen format | 4.2 | 4-12 |
| sample format | 3.4.2 | 3-15 | | 4.2.1 | 4-12 |
| use | 4.2.1 | 4-12 | dialog screens | 3.5 | 3-29 |
| | 4.2.1.2 | 4-16 | | 4.2 | 4-12 |
| | 3.4 | 3-12 | duplicate lines | 4.2.1.7 | 4-21 |
| | 3.4.2 | 3-15 | home screen | 4.4 | 4-29 |
| | | | | 3.2 | 3-2 |
| Conditional indicators | | | new format name and library | 4.1 | 4-2 |
| conditional values screen | 6.1.2.8 | 6-12 | optional screens | 3.2.2 | 3-4 |
| dialog screens | 3.4.1 | 3-13 | | 3.3.8 | 3-10 |
| | 3.5 | 3-28 | original formats | 4.2 | 4-12 |
| display properties | 3.5.4 | 3-40 | overlay formats | 3.3.4 | 3-8 |
| erase | 3.4.2 | 3-15 | | 3.3.4 | 3-8 |
| error message screen | 3.4.5 | 3-23 | | 4.3 | 4-25 |
| list screen | 6.1.2.6 | 6-9 | pass 1, basic format | 4.1.1 | 4-4 |
| optional screens | 3.4.1 | 3-13 | pass 1, complex format | 4.2.1.4 | 4-18 |
| replenish | 3.4.2 | 3-15 | pass 2, basic format | 4.1.2 | 4-6 |
| retain | 3.5.5 | 3-43 | pass 2, complex format | 4.2.1.5 | 4-18 |
| See also indicators. | | | pass 3, basic format | 4.1.3 | 4-9 |
| Conditional protection dialog (dialog screen 6) | 3.5.6 | 3-44 | pass 3, complex format | 4.2.1.6 | 4-19 |
| | | | screen format | 3.2.1 | 3-3 |
| | | | sequence of steps | 4.1 | 4-1 |
| | | | Currency sign | | |
| | | | automatically protected | 2.4 | 2-6 |
| | | | | 3.5.1 | 3-31 |
| | | | edit screen | 3.4.3 | 3-18 |
| | | | editing input or bidirectional fields | 1.3.2.3 | 1-11 |
| | | | fixed and floating insertion in numeric fields | 2.3.2 | 2-3 |

| Term | Reference | Page | Term | Reference | Page |
|----------------------------------|-----------|------|-------------------------------------|-------------|-----------|
| Cursor | | | | | |
| characteristics screen | 3.6 | 3-49 | | | |
| dialog screens | 3.5 | 3-29 | | | |
| | 3.6 | 3-51 | | | |
| home screen | 3.6 | 3-49 | | | |
| initial I/O screen | 4.1.3 | 4-9 | | | |
| optional screens | 3.6 | 3-51 | | | |
| positioning, creating a format | 3.6 | 3-49 | | | |
| positioning, filling in a format | 7.4 | 7-38 | | | |
| template screen | 4.1.1 | 4-4 | | | |
| type attribute screen | 4.1.2 | 4-7 | | | |
| | | | D | | |
| | | | DB character | | |
| | | | editing rules for signed output | | |
| | | | fields | Table 2-1 | 2-6 |
| | | | fixed insertion in numeric fields | 2.3.2 | 2-3 |
| | | | Decimal point | | |
| | | | automatic alignment | 2.4 | 2-5 |
| | | | dialog screen 1 | 3.5.1 | 3-34 |
| | | | editing input or bidirectional | | |
| | | | fields | 1.3.2.3 | 1-11 |
| | | | fixed insertion in numeric fields | 2.3.2 | 2-3 |
| | | | protection | 2.4 | 2-5 |
| | | | Default values | | |
| | | | characteristics screen | 4.1 | 4-2 |
| | | | | 4.2.1.1 | 4-15 |
| | | | dialog screen 2 | 3.5.2 | 3-35 |
| | | | dialog screen 4 | 3.5.4 | 3-40 |
| | | | dialog screens | 3.5.9 | 3-48 |
| | | | home screen | 3.2.2 | 3-4 |
| | | | | 4.1 | 4-2 |
| | | | | 4.2.1.1 | 4-15 |
| | | | | 5.1 | 5-2 |
| | | | I/O screen | 4.1.3 | 4-10 |
| | | | overwriting | 3.6 | 3-49 |
| | | | special display | 3.5.4 | 3-40 |
| | | | DELETE function | | |
| | | | home screen | 3.2 | 3-2 |
| | | | | 3.2.1 | 3-3 |
| | | | | 6.2 | 6-15 |
| | | | old format name and library | 3.2.3 | 3-5 |
| | | | use | 6.1 | 6-1 |
| | | | Device assignment set | See DVC-LFD | sequence. |
| | | | Dialog screen 1 | | |
| | | | changing I/O direction of field | 3.5.1 | 3-31 |
| | | | default values | 3.5.8 | 3-47 |
| | | | | 3.5.1 | 3-31 |
| | | | field requesting dialogs | 3.5.1 | 3-30 |
| | | | function | 3.5.1 | 3-30 |
| | | | | Table 3-2 | 3-29 |
| | | | internal length calculated by SFG | 3.5.1 | 3-32 |
| | | | rules | 3.5.1 | 3-35 |
| | | | sample format | 4.2.1 | 4-13 |
| | | | | 4.2.1.7 | 4-24 |
| | | | selecting other dialog screens | 3.5.1 | 3-33 |
| | | | type attributes and I/O direction | | |
| | | | of field | 3.5.1 | 3-33 |
| | | | | Table 3-3 | 3-34 |
| | | | type of internal usage | 3.5.1 | 3-31 |
| | | | warning messages on internal length | 3.5.1 | 3-32 |

| Term | Reference | Page | Term | Reference | Page |
|-----------------------------------|-----------|------|---|-----------|-------------------------|
| Dialog screen 2 | | | dialog screen 1 | 3.5.1 | 3-30 |
| alphabetic characters | 3.5.2 | 3-38 | dialog screen 2 | 3.5.2 | 3-35 |
| default values | 3.5.9 | 3-48 | dialog screen 3 | 3.5.3 | 3-38 |
| | 3.5.2 | 3-35 | dialog screen 4 | 3.5.4 | 3-40 |
| display of field | 3.5.2 | 3-36 | dialog screen 5 | 3.5.5 | 3-43 |
| field response | 3.5.2 | 3-36 | dialog screen 6 | 3.5.6 | 3-44 |
| function | 3.5.2 | 3-35 | dialog screen 7 | 3.5.7 | 3-45 |
| | Table 3-2 | 3-29 | dialog screen 8 | 3.5.8 | 3-42 |
| numeric values | 3.5.2 | 3-37 | duplicate lines | 4.4 | 4-29 |
| replenished input field | 3.5.2 | 3-35 | initiating | 4.1.3 | 4-9 |
| sample format | 4.2.1 | 4-12 | LIST function | 6.1.2 | 6-2 |
| | 4.2.1.7 | 4-24 | range checking | 3.5.8 | 3-46 |
| warning messages for line 3 | 3.5.2 | 3-36 | related dialog screens | Table 3-1 | 3-12 |
| | | | | 4.2 | 4-12 |
| Dialog screen 3 | | | related optional screens | Table 3-1 | 3-12 |
| default values | 3.5.9 | 3-48 | | 4.2 | 4-12 |
| function | 3.5.3 | 3-38 | special display properties | 3.5.4 | 3-40 |
| | Table 3-2 | 3-25 | use | 3.5 | 3-28 |
| | | | | 4.2 | 4-12 |
| Dialog screen 4 | | | | 4.2.1.6 | 4-19 |
| conditional display properties | 3.5.4 | 3-40 | | 4.2.1.7 | 4-21 |
| default display properties | 3.5.4 | 3-40 | Direction of fields | | |
| default values | 3.5.9 | 3-48 | dialog screen 1 | 3.5.1 | 3-30 |
| function | 3.5.4 | 3-40 | list screen | 6.1.2.6 | 6-9 |
| | Table 3-2 | 3-29 | RPG II | 7.3.1 | 7-5 |
| intensity/emphasis | 3.5.4 | 3-40 | variable data fields | 1.3.2.2 | 1-9 |
| | Table 3-4 | 3-42 | See also I/O directions, I/O fields, and bidirectional fields. | | |
| sample format | 4.2.1 | 4-12 | | | |
| | 4.2.1.7 | 4-21 | Display attributes | | See display properties. |
| Dialog screen 5 | | | | | |
| conditional retention | 3.5.5 | 3-43 | Display constants | | |
| | 3.4.6 | 3-24 | display properties | 3.4.4 | 3-20 |
| default values | 3.5.9 | 3-48 | emphasis | 3.4.4 | 3-20 |
| function | 3.5.5 | 3-43 | format layout | 4.1.1 | 4-4 |
| | Table 3-2 | 3-29 | intensity | 3.4.4 | 3-20 |
| Dialog screen 6 | | | part of screen format | 1.3.2.1 | 1-9 |
| conditional protection | 3.5.6 | 3-44 | protected and unprotected parts | 1.3.2.3 | 1-11 |
| default values | 3.5.9 | 3-48 | | | |
| function | 3.5.6 | 3-44 | Display control and error message | | |
| | Table 3-2 | 3-29 | display | 6.1.2.3 | 6-5 |
| Dialog screen 7 | | | | 6.1.2 | 6-2 |
| default values | 3.5.9 | 3-48 | Display functions | | |
| function | 3.5.7 | 3-45 | home screen | 3.2.1 | 3-3 |
| | Table 3-2 | 3-29 | LIST | 3.2.1 | 3-4 |
| range checking | 3.5.8 | 3-46 | | 3.2.3 | 3-4 |
| | | | | 6.1.2 | 6-1 |
| Dialogs | | | SHOW | 3.2.1 | 3-4 |
| changing field characteristics | 5.1 | 5-1 | | 3.2.3 | 3-5 |
| characteristics screen selections | Table 3-1 | 3-12 | SPOOL | 3.2.1 | 3-4 |
| conditional display | 3.5.3 | 3-38 | | 3.2.3 | 3-5 |
| conditional indicators | 3.4.1 | 3-13 | | 6.1.3 | 6-15 |
| | 3.5 | 3-28 | use | 3.2.1 | 3-3 |
| conditional protection | 3.5.6 | 3-44 | | 6.1 | 6-1 |
| conditional retention | 3.5.5 | 3-43 | | | |
| default values | 3.5.9 | 3-48 | | | |

| Term | Reference | Page | Term | Reference | Page |
|-----------------------------------|------------|------|---------------------------------------|-----------|------|
| Display properties | | | DMSEL,SCREEN (select a screen format) | | |
| alternate | Appendix E | | BAL imperative macroinstruction | 7.3.4 | 7-27 |
| conditional | 3.5.4 | 3-40 | BAL imperative macroinstruction | 7.3.4.4 | 7-32 |
| default | 3.5.4 | 3-40 | CD\$DDINF+11 error conditions | | |
| dialog screen 4 | 3.5.4 | 3-40 | returned | 7.3.4.4 | 7-33 |
| display constants | 3.4.4 | 3-20 | Duplicate lines | 4.4 | 4-29 |
| display control and error message | | | | | |
| display | 6.1.2.3 | 6-5 | DVC-LFD sequence | | |
| error message screen | 3.4.5 | 3.23 | COBOL | 7.3.2 | 7-10 |
| fields | 3.4.4 | 3-20 | examples | A.1 | A-3 |
| fields in error | 3.4.4 | 3-20 | job control | 7.2 | 7-1 |
| input fields | 3.4.4 | 3-20 | | | |
| output fields | 3.4.4 | 3-20 | | | |
| selections | Table 3-4 | 3-42 | | | |
| special display properties dialog | 3.5.4 | 3-40 | | | |
| special display screen | 3.4.4 | 3-20 | | | |
| Display retention | | | | | |
| characteristics screen | 3.3 | 3-6 | | | |
| display retention screen | 3.3.10 | 3-11 | | | |
| display retention screen | 3.4.6 | 3-24 | | | |
| Display retention screen | | | | | |
| characteristics screen | 3.3 | 3-6 | | | |
| characteristics screen | 3.3.10 | 3-11 | | | |
| field display attributes | 3.4.6 | 3-24 | | | |
| field display attributes | 3.5.4 | 3-40 | | | |
| field level display retention | 3.4.6 | 3-24 | | | |
| field level display retention | 3.5.5 | 3-43 | | | |
| use | 3.4 | 3-12 | | | |
| use | 3.4.6 | 3-24 | | | |
| DMINP (retrieve a record) | | | | | |
| BAL imperative macroinstruction | 7.3.4 | 7-27 | | | |
| BAL imperative macroinstruction | 7.3.4.3 | 7-30 | | | |
| CD\$DDINF+11 error codes returned | 7.3.4.3 | 7-31 | | | |
| DMOUT | 7.3.4.3 | 7-30 | | | |
| DMSEL | 7.3.4.3 | 7-30 | | | |
| function keys | 7.3.4.3 | 7-30 | | | |
| input record size | 7.3.4.3 | 7-30 | | | |
| response indicators | 7.3.4.3 | 7-30 | | | |
| variable-length records | 7.3.4.3 | 7-31 | | | |
| work area | 7.3.4.3 | 7-30 | | | |
| DMOUT (output a record) | | | | | |
| BAL imperative macroinstruction | 7.3.4 | 7-27 | | | |
| BAL imperative macroinstruction | 7.3.4.5 | 7-34 | | | |
| CD\$DDINF+11 error codes | 7.3.4.5 | 7-35 | | | |
| error conditions returned | 7.3.4.5 | 7-34 | | | |

| Term | Reference | Page | Term | Reference | Page |
|---|------------|------|---|-----------|------|
| E | | | | | |
| Edit characters | | | Erase indicator | 3.4.2 | 3-15 |
| edit screen | 3.4.3 | 3-18 | Erasing option | | |
| variable data fields | 4.1.2 | 4-6 | characteristics screen | 3.3 | 3-6 |
| | | | | 3.3.5 | 3-8 |
| | | | | A.2 | A-7 |
| Edit mask | | | conditional erase/replenish screen | 3.4.2 | 3-17 |
| dialog screen 1 | 3.5.1 | 3-30 | | 4.2.1 | 4-12 |
| editing input or bidirectional fields | 1.3.2.3 | 1-11 | Error message display | 6.1.2.3 | 6-5 |
| editing numeric fields | 2.4 | 2-4 | | | |
| Edit screen | | | Error message field, characteristics screen | 3.3 | 3-6 |
| rules for completion | 3.4.3 | 3-18 | | 3.3.9 | 3-11 |
| special editing characters (characteristics screen) | 3.3.7 | 3-10 | Error message screen | | |
| | 3.4.3 | 3-18 | characteristics screen | 3.4.5 | 3-21 |
| special editing display use | 6.1.2.2 | 6-5 | | 3.3.9 | 3-11 |
| | 3.4 | 3-12 | display control and error message display | 6.1.2.3 | 6-5 |
| | 3.4.3 | 3-18 | emphasis | 3.4.5 | 3-23 |
| yen sign for Katakana | 3.4.3 | 3-18 | intensity | 3.4.5 | 3-23 |
| | | | sample format | 4.2.1 | 4-12 |
| | | | | 4.2.1.3 | 4-17 |
| Editing | | | use | 3.4 | 3-12 |
| alphabetic and alphanumeric fields | 2.3.1 | 2-2 | | 3.4.5 | 3-21 |
| edit screen | 3.4.3 | 3-18 | Error retry counts, characteristics screen | 3.3 | 3-6 |
| examples | Table 2-2 | 2-7 | | 3.3.1 | 3-7 |
| insertion | 2.3.1 | 2-2 | | | |
| numeric fields | 2.3.2 | 2-3 | Errors | | |
| results | 2.4 | 2-4 | duplicate lines | 4.4 | 4-32 |
| rules for signed output fields | Table 2-1 | 2-6 | error message display | 6.1.2.3 | 6-5 |
| special characters | 3.3.7 | 3-10 | fields in | 3.4.4 | 3-20 |
| suppression/replacement | 2.3.2 | 2-4 | function key | 3.4.7 | 3-26 |
| variable data fields | 4.1.2 | 4-6 | indicated on workstation | 1.3.1 | 1-7 |
| | | | initial I/O screen | 4.1.3 | 4-10 |
| Editing characters | | | messages (home screen) | 3.7 | 3-51 |
| initiating dialogs | 4.1.3 | 4-10 | rules for dialog screens | 3.5 | 3-29 |
| input or bidirectional fields | 1.3.2.3 | 1-11 | type attribute screen | 4.1.2 | 4-9 |
| special | 3.3.7 | 3-10 | type of data | 1.4 | 1-13 |
| | 3.4.3 | 3-18 | ESCORT | 7.3.5 | 7-35 |
| Emphasis | | | Exclamation point | | |
| alternate display properties | Appendix E | | dialog screen 1 | 3.5.1 | 3-33 |
| device support | Appendix E | | editing alphabetic and alphanumeric fields | | |
| dialog screen 4 | 3.5.4 | 3-40 | | 2.3.1 | 2-2 |
| display constants | 3.4.4 | 3-20 | editing numeric fields | 2.3.2 | 2-3 |
| display property | 3.4.4 | 3-20 | initiating dialogs | 4.1.3 | 4-10 |
| | Table 3-4 | 3-42 | results of editing | 2.4 | 2-4 |
| error message screen | 3.4.5 | 3-23 | type attribute screen | 4.1.2 | 4-8 |
| fields in error | 3.4.4 | 3-20 | | | |
| input fields | 3.4.4 | 3-20 | | | |
| output fields | 3.4.4 | 3-20 | | | |
| selections | Table 3-4 | 3-42 | | | |
| special display properties | 3.5.4 | 3-40 | | | |
| special display screen | 3.4.4 | 3-20 | | | |
| | 3.5.4 | 3-40 | | | |
| English, default alphabet | 3.3.2 | 3-7 | | | |

| Term | Reference | Page | Term | Reference | Page |
|------------------------------------|-----------|------|---|-----------------------------|------|
| H | | | I | | |
| HELP function | | | I - for input, initial I/O screen | 4.1.3 | 4-9 |
| dialog screens | 3.5 | 3-29 | Imperative macroinstructions, BAL | See BAL. | |
| prompting by SFG | 1.3.1 | 1-7 | IMS | | |
| requesting | 3.8 | 3-53 | programming considerations | 7.3.6 | 7-36 |
| Home screen | | | sample COBOL program | Appendix F | |
| CREATE-FROM function | 3.2.1 | 3-3 | Indicators | | |
| | 3.2.2 | 3-5 | BAL | 7.3.4.3 | 7-30 |
| | 5.1 | 5-1 | COBOL | 7.3.2.2 | 7-18 |
| | 5.2 | 5-6 | conditional | See conditional indicators. | |
| CREATE function | 3.2.1 | 3-2 | display retention | 3.4.6 | 3-24 |
| | 3.2.2 | 3-5 | erase | 3.4.2 | 3-17 |
| | 3.3.2 | 3-7 | option | 3.4.7 | 3-26 |
| | 4.1 | 4-1 | | 7.3.2.2 | 7-18 |
| cursor positioning and overwriting | 3.6 | 3-49 | protection | 3.5.6 | 3-44 |
| DELETE function | 3.2.1 | 3-3 | replenish | 3.4.2 | 3-15 |
| | 3.2.3 | 3-5 | response | 3.4.7 | 3-26 |
| | 6.2 | 6-15 | | 7.3.2.2 | 7-18 |
| display functions | 6.1 | 6-1 | retain | 7.3.4.3 | 7-30 |
| functions provided by SFG | 3.2.1 | 3-3 | | 3.5.3 | 3-39 |
| LIST function | 3.2.1 | 3-4 | | 3.5.5 | 3-43 |
| | 3.2.3 | 3-5 | | 3.5.6 | 3-44 |
| | 6.1.2 | 6-1 | RPG II | 3.4.7 | 3-27 |
| MODIFY function | 3.2.1 | 3-3 | special display properties | 3.5.4 | 3-41 |
| | 3.2.3 | 3-5 | syntax | 3.4.1 | 3-13 |
| | 5.1 | 5-1 | | 3.5.3 | 3-38 |
| new format name and library | 3.2.2 | 3-4 | | 3.5.4 | 3-41 |
| old format name and library | 3.2.3 | 3-5 | | 3.5.5 | 3-43 |
| sample format | 4.2.1.1 | 4-15 | | 3.5.6 | 3-44 |
| SHOW function | 3.2.1 | 3-3 | Initial I/O screen | 4.1.3 | 4-9 |
| | 3.2.3 | 3-5 | Input data | | |
| | 6.1.1 | 6-1 | transfer by screen format | | |
| SPOOL function | 3.2.1 | 3-4 | coordinator | 1.2 | 1-4 |
| | 3.2.3 | 3-5 | variable data fields | 1.3.2.2 | 1-9 |
| | 6.1.3 | 6-15 | Input fields | See I/O fields. | |
| TERMINATE function | 3.2.1 | 3-4 | Input format specifications form (RPG II) | 7.3.1 | 7-5 |
| | 6.2 | 6-15 | | Fig. 7-1 | 7-7 |
| | | | | 7.3.6 | 7-36 |
| | | | Input record screen | 6.1.2 | 6-2 |
| | | | | 6.1.2.9 | 6-14 |
| | | | Insertion editing | | |
| | | | alphabetic and alphanumeric fields | 2.3.1 | 2-2 |
| | | | dialog screen 1 | Table 3-2 | 3-29 |
| | | | numeric fields | 3.5.1 | 3-30 |
| | | | | 2.3.2 | 2-3 |

| Term | Reference | Page | Term | Reference | Page |
|------------------------------------|-----------|------|-------------------------------------|-----------|------|
| L | | | M | | |
| Language, characteristics screen | 3.3.2 | 3-7 | Minus sign | | |
| Library file (user-assigned) | | | editing rules for signed output | | |
| DVC/LFD sequence | 7.2 | 7-1 | fields | Table 2-1 | 2-6 |
| modifying formats | A.1 | A-3 | fixed and floating insertion in | | |
| | 5.1 | 5-1 | numeric fields | 2.3.2 | 2-3 |
| LIST function | | | MIRAM user library, storing formats | 1.2 | 1-4 |
| conditional values screen | 6.1.2.8 | 6-12 | MODIFY function | | |
| display control and error message | | | characteristics screen | 3.3.2 | 3-7 |
| display | 6.1.2.3 | 6-5 | dialog screen 1 | 3.5.1 | 3-30 |
| format display | 6.1.2.5 | 6-7 | formats | 3.2.1 | 3-3 |
| function/command key display | 6.1.2.4 | 6-6 | | 3.3.2 | 3-7 |
| home screen | 3.2 | 3-2 | | 3.3.4 | 3-8 |
| | 3.2.1 | 3-3 | | 5.1 | 5-1 |
| | 3.2.3 | 3-5 | home screen | 3.2 | 3-2 |
| input record screen | 6.1.2.9 | 6-14 | | 3.2.1 | 3-3 |
| list screen | 6.1.2.6 | 6-7 | | 3.2.3 | 3-5 |
| old format name and library | 3.2.3 | 3-5 | language | 3.3.2 | 3-7 |
| output record screen | 6.1.2.10 | 6-14 | modify screen | 5.1.1 | 5-2 |
| special editing display | 6.1.2.2 | 6-5 | old format name and library | 3.2.3 | 3-5 |
| sublist screen | 6.1.2.7 | 6-10 | original and overlay formats | 3.3.4 | 3-8 |
| summary screen | 6.1.2.1 | 6-3 | summary | 5.2 | 5-6 |
| use | 3.2.1 | 3-3 | | | |
| | 6.1 | 6-1 | Modify screen | | |
| | 6.1.2 | 6-1 | completing | 5.1.1 | 5-2 |
| List screen | 6.1.2 | 6-2 | dialog screen 1 | 3.5.1 | 3-30 |
| | 6.1.2.6 | 6-6 | modification summary | 5.2 | 5-6 |
| | 6.1.2.7 | 6-9 | option 1 (CHANGE TEMPLATE) | 5.1.1.1 | 5-3 |
| Logon procedure | | | option 2 (CHANGE TYPE) | 5.1.1.2 | 5-4 |
| activating the SFG | 3.1 | 3-1 | option 3 (CHANGE I/O) | 5.1.1.3 | 5-4 |
| creating screen format | 4.1 | 4-1 | option 7 (CHANGE TEMPLATE ONLY - | | |
| sample format | 4.2.1 | 4-12 | NO INTERNAL CHANGES) | 5.1.1.4 | 5-4 |
| Lowercase translation | | | | | |
| characteristics screen | 3.3 | 3-6 | | | |
| | 3.3.3 | 3-7 | | | |
| range checking | 3.5.8 | 3-48 | | | |
| Low-intensity display | | | | | |
| protected and unprotected parts of | | | | | |
| screen format | 1.3.2.3 | 1-11 | | | |
| special display screen | 3.4.4 | 3-20 | | | |

| Term | Reference | Page | Term | Reference | Page |
|---|---------------|--------------|--|----------------------------|--------------------|
| N | | | O | | |
| New format name and library, home screen | 3.2 3.2.2 | 3-2 3-4 | O - for output, I/O screen | 4.1.3 | 4-9 |
| Nondisplay screen | | | Old format name and library, home screen | 3.2 3.2.3 | 3-2 3-4 |
| characteristics screen | 3.3.12 | 3-11 | OPEN (open a file), BAL imperative macroinstruction | 7.3.4.1 | 7-28 |
| use | 3.4 3.4.8 | 3-12 3-27 | Option indicators | | |
| Nondisplayed field | | | COBOL | 7.3.2.2 | 7-18 |
| characteristics screen | 3.3 3.3.12 | 3-6 3-12 | function/command key screen | 3.4.7 | 3-26 |
| nondisplay screen | 3.4.8 | 3-27 | Optional screens | | |
| Numeric fields | | | characteristics screen | Table 3-1 | 3-12 |
| default values for dialog screens | 3.5.9 | 3-48 | conditional erase/replenish screen | 3.4.2 | 3-15 |
| editing | 2.3.2 | 2-3 | conditional indicators | 3.4.1 | 3-13 |
| range checking | 3.5.8 | 3-46 | dialog screens | Table 3-1 | 3-12 |
| rules | 2.2 | 2-1 | display retention screen | 3.4.6 | 3-24 |
| type attributes | 4.1.2 | 4-6 | edit screen | 3.4.3 | 3-18 |
| variable data fields | 1.3.2.2 | 1-9 | error message screen | 3.4.5 | 3-21 |
| 9 character | | | function/command key screen | 3.4.7 | 3-26 |
| indicating alphanumeric fields | 2.2 | 2-2 | nondisplay screen | 3.4.8 | 3-27 |
| indicating numeric fields | 2.2 | 2-2 | special display screen | 3.4.4 | 3-20 |
| | | | use | 3.4 | 3-12 |
| | | | Original format, characteristics screen | 3.3 3.3.4 4.1 | 3-6 3-8 4-2 |
| | | | Output data, transfer by SFC | 1.2 | 1-4 |
| | | | Output fields | See I/O fields. | |
| | | | Output format specifications form (RPG II) | 7.3.1 Fig. 7-1 7.3.6 | 7-6 7-8 7-37 |
| | | | Output record screen | 6.1.2 6.1.2.10 | 6-2 6-14 |
| | | | Overlay formats | | |
| | | | advantage | 4.3.1 | 4-25 |
| | | | characteristics screen | 3.3.4 | 3-8 |
| | | | creation | 4.3.2 | 4-28 |
| | | | SOE character | 4.3.2 | 4-28 |
| | | | use | 4.3 4.3.1 | 4-25 4-25 |

| Term | Reference | Page | Term | Reference | Page |
|------------------------------------|---------------------------|------|-----------------------------------|----------------------|------|
| S | | | | | |
| Screen format coordinator | | | duplicate lines | 4.4 | 4-29 |
| BAL | 7.3.4.5 | 7-34 | edit screen | 3.4.3 | 3-18 |
| COBOL | 7.3.2.2 | 7-18 | erasing option | See erasing option. | |
| conditional erase/replenish screen | 3.4.2 | 3-16 | error and warning messages | 3.7 | 3-51 |
| conditional indicators | 3.4.1 | 3-13 | error message field | 3.3.9 | 3-11 |
| dialog screen 2 | 3.5.2 | 3-35 | error message screen | 3.4.5 | 3-21 |
| dialog screen 5 | 3.5.5 | 3-43 | error retry counts | 3.3.1 | 3-7 |
| display retention | 3.3.10 | 3-11 | format display | 6.1.2.5 | 6-7 |
| display retention screen | 3.4.6 | 3-24 | function | 1.2 | 1-4 |
| edit screen | 3.4.3 | 3-19 | | Fig. 1-4 | 1-5 |
| error message screen | 3.4.5 | 3-22 | | 1.3 | 1-6 |
| FORTRAN | 7.3.3 | 7-26 | | 3.2.1 | 3-3 |
| function | 1.2 | 1-4 | | 5.1 | 5-1 |
| | Fig. 1-4 | 1-5 | | 6.1 | 6-1 |
| | 1.4 | 1-13 | | 6.2 | 6-15 |
| function/command key screen | 3.4.7 | 3-26 | function/command key display | 6.1.2.4 | 6-6 |
| job control | 7.2 | 7-1 | function/command key screen | 3.4.7 | 3-26 |
| option indicators | 3.4.7 | 3-26 | function keys | 3.3.11 | 3-11 |
| program considerations | 7.3 | 7-5 | HELP function | 3.8 | 3-53 |
| programmer responsibilities | 7.1 | 7-1 | help screens | Appendix B | |
| range checking | 3.5.8 | 3-47 | home screen | 3.2 | 3-1 |
| REPLENISH SCREEN option | 3.3.5 | 3-9 | initiating dialogs | 4.1.3 | 4-9 |
| response indicators | 3.4.7 | 3-26 | input record screen | 6.1.2.9 | 6-14 |
| RPG II | 7.3.1 | 7-5 | LIST function | See LIST function. | |
| software element of screen | | | list screen | 6.1.2.6 | 6-7 |
| format services | 1.2 | 1-4 | lowercase translation | 3.3.1 | 3-7 |
| UNLOCK KEYBOARD option | 3.3.4 | 3-8 | MODIFY function | See MODIFY function. | |
| workstation considerations | 7.3.2.4 | 7-26 | | 5.1 | 5-1 |
| | 7.4 | 7-38 | modifying formats | 5.2 | 5-6 |
| | | | | 3.2.2 | 3-4 |
| Screen format generator | | | new format name and library | 3.4.8 | 3-27 |
| activating | 3.1 | 3-1 | nondisplay screen | 3.3.12 | 3-12 |
| alphabet | 3.3.2 | 3-7 | nondisplayed field | 3.2.3 | 3-5 |
| basic screen formats | 4.1 | 4-1 | optional screens | 3.4 | 3-12 |
| characteristics screen | 3.3 | 3-6 | | 4.2 | 4-12 |
| complex screen formats | 4.2 | 4-12 | original formats | 3.3.4 | 3-8 |
| conditional erase/replenish screen | 3.4.2 | 3-15 | | 4.1 | 4-2 |
| conditional indicators | 3.4.1 | 3-13 | output record screen | 6.1.2.10 | 6-14 |
| conditional values screen | 6.1.2.8 | 6-12 | overlay formats | 3.3.4 | 3-8 |
| CREATE function | See CREATE function. | | | 4.3.2 | 4-28 |
| | | | | 4.3.1 | 4-25 |
| CREATE-FROM function | See CREATE-FROM function. | | passes | See passes. | |
| | | | prompting | 1.3.1 | 1-7 |
| cursor positioning and overwriting | 3.6 | 3-49 | sample format | 4.2.1 | 4-12 |
| default values for dialog screens | 3.5.9 | 3-48 | SHOW function | See SHOW function. | |
| DELETE function | See DELETE function. | | | | |
| | | | software element of screen format | | |
| dialog screens | 3.5 | 3-28 | services | 1.2 | 1-4 |
| | 4.2 | 4-12 | special display control | 3.3.8 | 3-10 |
| display control and error message | | | special display screen | 3.4.4 | 3-20 |
| display | 6.1.2.3 | 6-5 | special editing characters | 3.3.7 | 3-10 |
| display functions | See display functions. | | special editing display | 6.1.2.2 | 6-5 |
| | | | SPOOL function | See SPOOL function. | |
| display retention | 3.3.10 | 3-11 | | | |
| display retention screen | 3.4.6 | 3-24 | | | |

| Term | Reference | Page | Term | Reference | Page |
|------------------------------------|-----------------------------------|------|--|--------------------------------------|------|
| Screen format generator (cont) | | | overlay | 3.3.4 | 3-8 |
| sublist screen | 6.1.2.7 | 6-10 | | 4.3 | 4-25 |
| summary screen | 6.1.2.1 | 6-3 | program considerations | 7.3 | 7-5 |
| TERMINATE function | See TERMINATE function. | | program use | 7.1 | 7-1 |
| unlocking option | 3.3.5 | 3-8 | programmer responsibilities | 7.1 | 7-1 |
| | 3.4.2 | 3-16 | prompting | 1.3.1 | 1-7 |
| | 4.1 | 4-3 | protected and unprotected parts | 1.3.2.3 | 1-11 |
| | 4.2.1.1 | 4-15 | RPG II | 7.3.1 | 7-5 |
| | | | sample | 4.2.1 | 4-12 |
| | | | SYSRES volume | 1.2 | 1-4 |
| Screen format services | | | system library (\$Y\$FMT) | 1.2 | 1-4 |
| format creation | 1.2 | 1-1 | template screen | Fig. 4-1 | 4-4 |
| format use | 1.2 | 1-1 | type attribute screen | Fig. 4-2 | 4-7 |
| interactive component of OS/3 | 1.2 | 1-1 | type attributes and editing | 4.1.2 | 4-6 |
| screen format coordinator (SFG) | See screen format coordinator. | | variable data fields | 1.3.2.2 | 1-9 |
| screen format generator (SFG) | See screen format generator. | | SFC | See screen format coordinator. | |
| software elements | 1.2 | 1-4 | | | |
| Screen formats | | | SFG | See screen format generator. | |
| BAL | 7.3.4 | 7-27 | | | |
| basic | 4.1 | 4-1 | SFS | See screen format services. | |
| characteristics screen | See characteristics screen. | | | | |
| COBOL | 7.3.2 | 7-9 | SHOW function | | |
| complex | 4.2 | 4-12 | home screen | 3.2 | 3-1 |
| components | 1.3.2 | 7-9 | | 3.2.1 | 3-3 |
| creating | 1.3.1 | 1-7 | old format name and library use | 3.2.3 | 3-5 |
| cursor positioning and overwriting | 3.6 | 3-49 | | 3.2.1 | 3-3 |
| deleting | 1.3.1 | 1-7 | | 6.1 | 6-1 |
| | 6.2 | 6-15 | | 6.1.1 | 6-1 |
| | 3.2.1 | 3-3 | Signed output fields, editing rules | Table 2-1 | 2-6 |
| | 3.2.3 | 3-5 | | | |
| dialogs | 3.5 | 3-28 | Simple insertion, editing numeric fields | 2.3.2 | 2-3 |
| display constants | 1.3.2.1 | 1-9 | | | |
| displaying | 1.2 | 1-1 | Slash, dialog screen 1 | 3.5.1 | 3-34 |
| | 1.3.1 | 1-7 | | | |
| duplicate lines | 4.4 | 4-29 | SOE character, overlay format | 4.3.2 | 4-28 |
| error and warning messages | 3.7 | 3-51 | | | |
| error indication | 1.3.1 | 1-7 | Special display control | | |
| ESCORT | 7.3.5 | 7-35 | characteristics screen | 3.3 | 3-6 |
| FORTRAN | 7.3.3 | 7-26 | | 3.3.8 | 3-10 |
| home screen | See home screen. | | special display screen | 3.4.4 | 3-20 |
| input | 1.2 | 1-1 | | | |
| job control | 7.2 | 7-1 | Special display screen | | |
| layout | 1.2 | 1-4 | display control and error message display | 6.1.2.3 | 6-5 |
| | 4.1.1 | 4-4 | display properties | 3.4.4 | 3-20 |
| MIRAM user library | 1.2 | 1-4 | | Appendix E | |
| modifying | 1.3.1 | 1-7 | use | 3.4 | 3-12 |
| | 3.2.1 | 3-3 | | 3.4.4 | 3-20 |
| | 5.1 | 5-1 | | | |
| original | 3.3.4 | 3-8 | | | |
| | 4.1 | 4-1 | | | |
| output | 1.2 | 1-1 | | | |

| Term | Reference | Page | Term | Reference | Page |
|---|-----------|------|---|-----------|------|
| | | | W | | |
| Unlocking option | | | | | |
| characteristics screen | 3.3 | 3-6 | | | |
| | 3.3.5 | 3-8 | Warning messages | 3.7 | 3-51 |
| conditional erase/replenish screen | 3.4.2 | 3-17 | | | |
| creating basic format | 4.1 | 4-3 | Workstation considerations | | |
| creating complex format | 4.2.1.1 | 4-16 | interprogram communication | 7.3.2.4 | 7-26 |
| | | | operator | 7.4 | 7-38 |
| Unprotected fields | | | | | |
| description | 1.3.2.3 | 1-11 | | | |
| SHOW function | 6.1.1 | 6-1 | | | |
| Uppercase alphabetical characters, translation | 3.3.1 | 3-6 | | | |
| USE SFS job control statement | 7.2 | 7-1 | X | | |
| | A.1 | A-1 | X character, indicating alphanumeric fields | 2.2 | 2-2 |
| | | | Y | | |
| V | | | | | |
| Variable data fields | | | | | |
| considerations | 2.1 | 2-1 | Yen sign, default currency sign for Katakana | 3.4.3 | 3-18 |
| dialog screens | 3.5 | 3-28 | | | |
| | 3.5.1 | 3-30 | | | |
| display retention screen | 3.4.6 | 3-24 | | | |
| duplicate lines | 4.4 | 4-29 | | | |
| edit characters | 4.1.2 | 4-6 | | | |
| IMS | 7.3.6 | 7-36 | | | |
| I/O direction | 4.2.1.6 | 4-19 | | | |
| layout | 1.3.2.2 | 1-9 | | | |
| | 4.2.1.4 | 4-18 | | | |
| pass 1 | 4.1.1 | 4-4 | | | |
| | 4.2.1.4 | 4-18 | | | |
| pass 2 | 4.1.2 | 4-6 | | | |
| | 4.2.1.5 | 4-18 | | | |
| pass 3 | 4.1.3 | 4-9 | Z | | |
| | 4.2.1.6 | 4-19 | Z character, suppression/replacement editing in numeric fields | 2.3.2 | 2-3 |
| protected and unprotected parts | 1.3.2.3 | 1-11 | | | |
| RPG II | 7.3.1 | 7-5 | Zero character | | |
| type attributes | 2.2 | 2-1 | editing alphabetic and alphanumeric fields | 2.3.2 | 2-3 |
| | 4.1.2 | 4-6 | insertion in numeric fields | 2.3.2 | 2-3 |
| | 4.2.1.5 | 4-18 | | | |
| See also bidirectional fields and I/O fields. | | | | | |





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

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



USER COMMENTS

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 Level)

Comments:

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage is necessary if mailed in the U.S.A.)
Thank you for your cooperation

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: SYSTEM PUBLICATIONS



P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19422-9990



FOLD



USER COMMENTS

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 Level)

Comments:

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage is necessary if mailed in the U.S.A.)
Thank you for your cooperation



FOLD



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

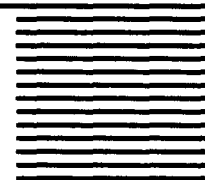
BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY CORPORATION

**ATTN: Documentation Quality Control Group
C/O SYSTEM PUBLICATIONS**



P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19422-9990



FOLD