Burroughs

User's Guide

B 1000 Series

Generalized Message Control System (GEMCOS) Format Generator

(Relative to the 7.0 Software Release)

Priced Item
Printed in U.S.A.
February 1985

1164019

Burroughs

User's Guide

B 1000 Series Generalized Message Control System (GEMCOS) Format Generator

(Relative to the 7.0 Software Release)

Priced Item
Printed in U.S.A.
February 1985

1164019

Burroughs cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Comments or suggestions regarding this document should be submitted on a Field Communication Form (FCF) with the Class specified as "2" (System Software), the Type specified as "1" (F.T.R.), and the Product specified as the seven-digit form number of the manual (for example, "1164019"). The FCF should be sent to the following address:

Burroughs Corporation
Product Assurance and Support
3519 W. Warner Avenue
Santa Ana, CA 92704

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

INTRODUCTION

The User's Guide, B 1000 Series Generalized Message Control System (GEMCOS) Format Generator is written for Burroughs customer programmers who are writing application programs that require message formatting in an Advanced or Total GEMCOS environment.

The guide contains instructions for the operation of the system's two format creation programs. It also covers the syntax and defaults of the commands used by the two programs.

The B 1000 Series Format Generator is available on Burroughs B 1000 computers under the following style identification: B1000 MCF.

Section 1 of this manual provides an overview of the GEMCOS Format Generator System. Section 2 describes a nonintepretive formatting session using the Format Maker. Section 3 illustrates an intepretive formatting session with the Format Generator. A complete description of all Format Maker and Format Generator commands is included in Section 4. The following are included in the appendices for each of the format creation programs:

- The names and purposes of the B 1000 files used in format creation.
- The system limits and defaults.
- The error messages.

The information in this manual can be supplemented from the following manuals:

- User's Guide, B 1000 Series Generalized Message Control System (GEMCOS) (Relative to 7.0 Software Release), form 1163920.
- B 1800/B 1700 Generalized Message Control System (GEMCOS) Formatting Guide, form 1106531.

- Capabilities Manual, B 1000 Series Generalized Message Control System (GEMCOS), (Relative to 7.0 Software Release), form 1164001.

- B 1000 System Software Operation Guide, form 1151982.

SECTION 1

SYSTEM OVERVIEW

The B 1000 Generalized Message Control System (GEMCOS) Format Generator product consists of two mutually independent formatting programs, one interpretive (Format Generator) and one noninterpretive (Format Maker). These programs are designed to free the GEMCOS programmer from most of the work involved in creating GEMCOS formats.

Each program allows formats to be laid out (entered as literal text to create a screen image), tested, and stored. And each supplies updated formats to the "live" GEMCOS network from an on-line terminal without requiring that GEMCOS be brought down or that other processing be disturbed.

Both programs are designed to run in the following environments:

- Under the B 1000 GEMCOS MCS.
- As free-standing application programs.
- Without the supervision of an MCS.

Note that if either program is run under the control of a GEMCOS MCS, it should be declared in the TCL as either a nonparticipating UTILITY or an ASSIGNMENT program.

FORMAT MAKER PROGRAM

The Format Maker program creates and maintains noninterpretive B 1000 GEMCOS on-line terminal formats interactively. The user can add, change, and delete formats in the Test Format file, then "update" them in the noninterpretive Live Format file. Because a noninterpretive format is stored essentially as it is seen on the terminal screen, formatted output messages can be presented with very little delay.

Figure 1-1 shows a diagram of the processes and actions taken by the Format Maker.

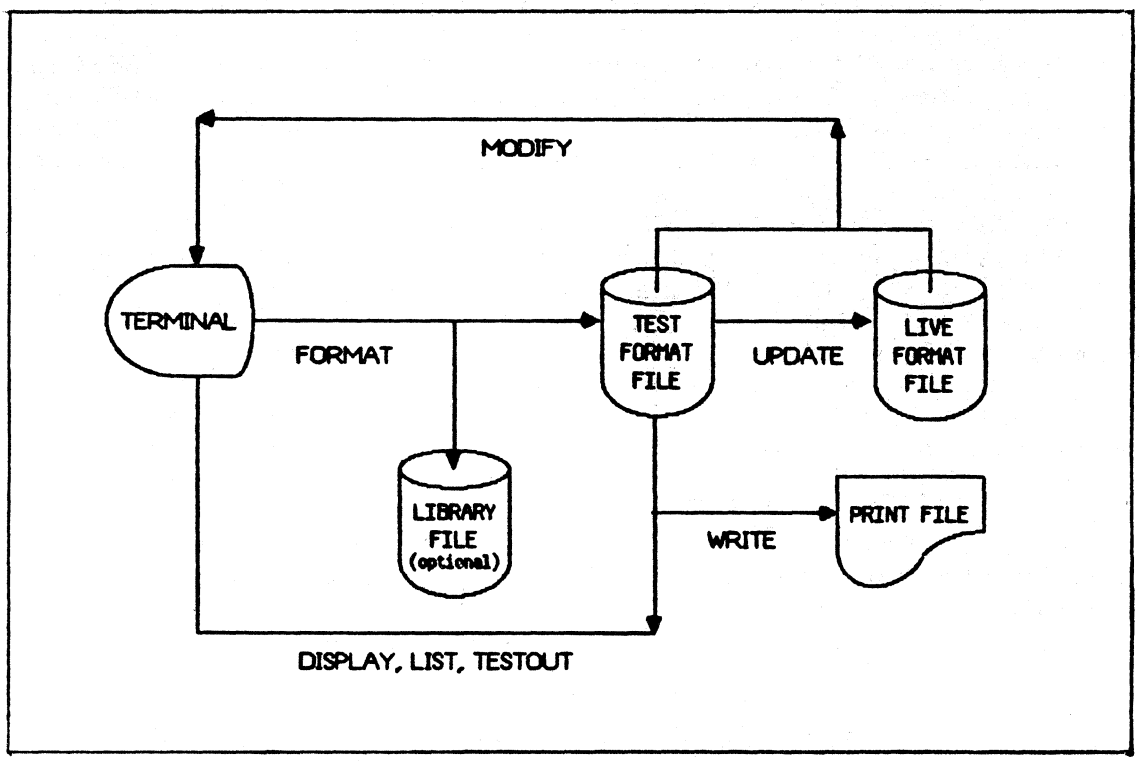


Figure 1-1. Format Maker Diagram

FORMAT MAKER FORMATS

A Format Maker format describes how the parts of a message should be displayed on output to the terminal device. Because formats are laid out on the terminal screen, the user can create the necessary layouts without knowledge of the devices that send the data. However, since the Format Maker formats are stored as TD830 cursor-positioning strings, Format Maker output formats can only be created or used on a TD830-compatible terminal device.

FORMAT GENERATOR PROGRAM

The Format Generator program, like Format Maker, allows the user to lay out formats on an on-line B 1000 GEMCOS terminal screen. However, since Format Generator stores its formats in the form of TCL formatting codes, formats can be designed for specific terminal device types. This is done simply by running the Format Generator program from an on-line terminal of the required type.

The Format Generator program frees the GEMCOS programmer from concern with Transaction Control Language (TCL) requirements when describing simple formats. It also assumes most of the work required for creating complicated formats. At the end of a formatting session, the Format Generator program allows the user to save the TCL formatting specifications generated by the program during that session.

Figure 1-2 shows a diagram of the processes and actions taken by the Format Generator.

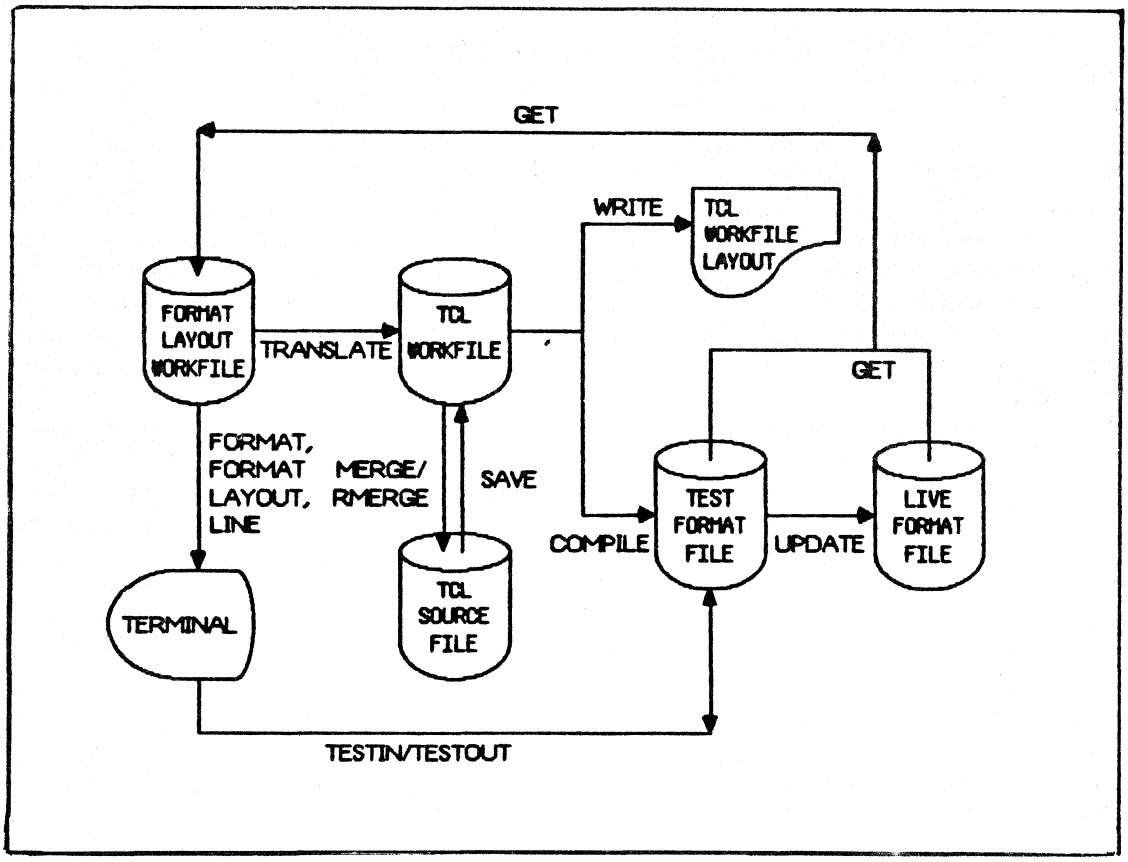


Figure 1-2. Format Generator Diagram

FORMAT GENERATOR FORMATS

A Format Generator format describes how the parts of a message should be handled from a terminal device (on input) or from an application (on output). Typically, this handling includes expanding, compressing, inserting, deleting, and rearranging fields, and translating data.

A Format Generator format can also contain integrity checks, such as tests for all-numeric characters. On output, the programming features of the terminal device can be used to create more readable displays and to send device-independent data structures.

Because the format layout is created by "painting," programmers can define their message layouts without knowledge of the devices which send and receive data. The appearance of the final message can be determined by a

format, allowing the application programmer to disregard differences between terminals.

The Format Generator builds a format type called a DISPLAY format. Its main function is to produce more readable program output. The DISPLAY format allows the application programmer to send out a device-independent data structure. The Format Generator then adds titles and rearranges data to create the terminal display. Without formatting, programs must specify arrangements for spacing, titles, and forms control.

SELECTING THE APPROPRIATE PROGRAM

Although the Format Maker is similar to the Format Generator, there are some important differences between the two. For example, because Format Generator interacts as it does with the TCL, it can create formats capable of inserting, substituting, or rearranging fields as data strings are fit into the pre-designed formats. Format Maker formats, being noninterpretive, offer a simpler, quicker alternative with a faster response time.

The following differences should be considered in deciding whether a given format will be created using Format Maker or Format Generator:

- Format Maker allows one user at a time, while Format Generator allows as many as 10 users on one copy of the program.
- Format names used by the Format Maker can only be six characters long. Format Generator names may be as long as 17 characters.
- Format Maker formats cannot be described in the TCL. The format name must function as the message ID for that format. Format Generator formats can be described in the TCL.
- A Format Maker format must be created and used on a TD830-compatible terminal. A Format Generator format can be used on whatever device type was used in its creation.

- The status line (line 25) of the terminal screen is used by the Format Maker, but not by the Format Generator.

- Format Maker formats are processed faster than Format Generator formats.

USING INFORMATIONAL COMMANDS

Both formatting programs offer information to the user through the following commands:

- HELP
- STATUS
- TEACH
- WHAT

There are essentially only two different commands in this category, since HELP and TEACH are synonymous, as are STATUS and WHAT. The HELP and TEACH commands inform the user of available commands and their syntax. The STATUS and WHAT commands are used to obtain information about a particular station that is attached to the formatting program. Syntax information on each command is in Sections 3 and 5.

SECTION 2

FORMAT MAKER FORMATTING SESSION

The Format Maker program creates noninterpretive GEMCOS output formats for use by a TD830-compatible terminal. Formats are stored as packed TD830 cursor-positioning strings. Because Format Maker formats are not designed to interact with the TCL, they cannot insert, substitute, or rearrange fields, as the Format Generator program can. However, because formats are already in a form usable by the TD830, the response time is reduced.

A formatting session with its required commands is described in this section, including the following tasks:

1. Initiating the program.
2. Creating or modifying formats.
3. Testing formats.
4. Updating format files.
5. Saving or purging the work file.
6. Activating new formats.

FORMAT MAKER FILES

The following files are used by Format Maker in a formatting session:

- Test Format file.
- Live Format file.
- Library file.

The Test Format file contains all the formats created or modified during the current formatting session. It is either saved or purged at the end of the session as specified by the user.

One or more formats from the Test Format file can be transferred by Format Maker to the Live Format file for use by the MCS. The Live Format file can be updated at any time without requiring that the MCS be brought down.

The Library Format file is a COBOL library containing the COBOL entries corresponding to each format field for which LIBRARY has been set equal to TRUE. These entries can be merged by the COBOL programmer into an application program.

INITIATING FORMAT MAKER

The Format Maker program can be initiated in one of two ways:

- If not running under the control of a GEMCOS MCS, Format Maker is executed in the same manner that any application program is executed:

  ```
  EX GEMCOS/FORMMAKER <optional file equates>
  ```

- If Format Maker is running under the control of a GEMCOS MCS, the GEMCOS EX network control command is entered from the user's station, and the following must be declared in the TCL.

  ```
  MAXASSIGNERS  =  1.
  INTERFACE     =  NONPARTICIPATION.
  ```

Note that the terminal should not be in scroll mode when a formatting session is initiated. "?-" must be entered in home position to take the terminal out of scroll mode.

The BYE command ends the formatting session and causes the Test Format file to be either saved or purged, as the user specifies. If no option is specified and the Test Format file is empty, Format maker will purge the file. If no option is specified and the Format Test file contains one or more files, the file will be saved. When Format Maker reaches end-of-job, it will list the following:

- Test Format file name.
- Whether it was saved or purged.
- Library Format File name.

BEGINNING A SESSION

After Format Maker is initiated, a formatting session begins with the definition of the character to be used as the right forms mode delimiter. The default for this is the RS character (ASCII 30). If the WRITE command is to be used to produce a printed copy of a format, the right delimiter character should match the contents of the TD830 register 87. The forms mode right delimiter character becomes important for such features as bright highlighting or reverse video. See "Modifying a Format" later in this section.

In the first screen of Figure 2-1, the program requests the delimiter character to be used. In the second screen, the user enters the delimiter character.

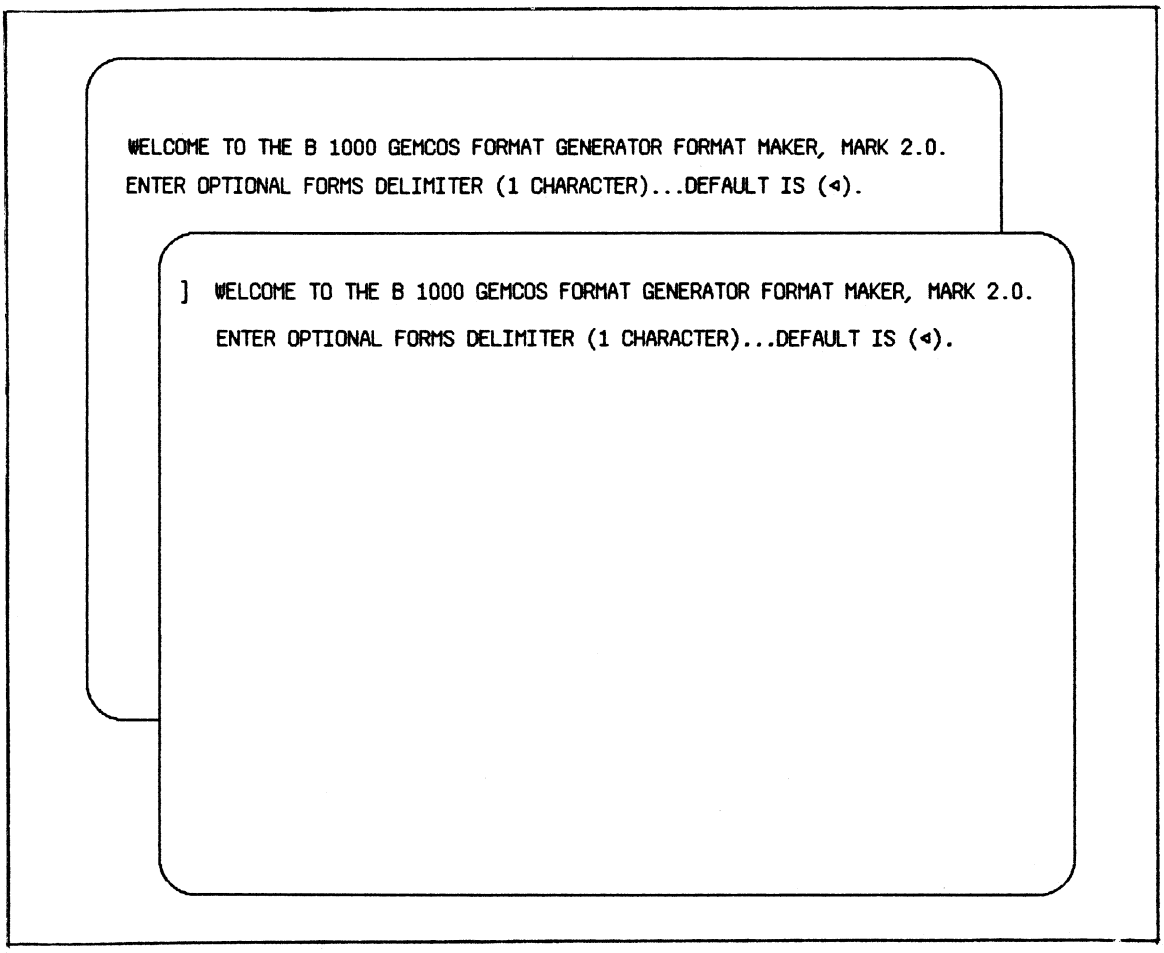


Figure 2-1. Assigning the Right Delimiter.

Once the delimiter character is defined, Format Maker sends a message indicating the start of the session. The program then waits for the receipt of the first command.

Commands are entered in a free-form manner from anywhere on the terminal screen. However, if a command is not entered at home position, it must be preceded only by blanks. When the terminal is in Expecting Format Layout mode (explained in this section under "Creating a Format"), commands should be entered starting at the first character position on the screen.

CREATING A FORMAT

To create a format, the FORMAT command must be issued. See Section 3 for syntax information. The FORMAT command places the terminal in Expecting Format Layout mode. When in this mode, the user can lay out the format as it should appear, or cancel the FORMAT command by entering IGNORE.

Figure 2-2 illustrates the user's entry of the FORMAT command and the response by Format Maker. In the first screen of this figure, the format to be defined is named Custmr (six characters, as required by Format Maker). This format will be used to illustrate each of the steps discussed in this section. Note that the Custmr format is assigned the LIBRARY attribute. The default value for the LIBRARY attribute is FALSE. See "FORMAT" in Section 3 for more information.

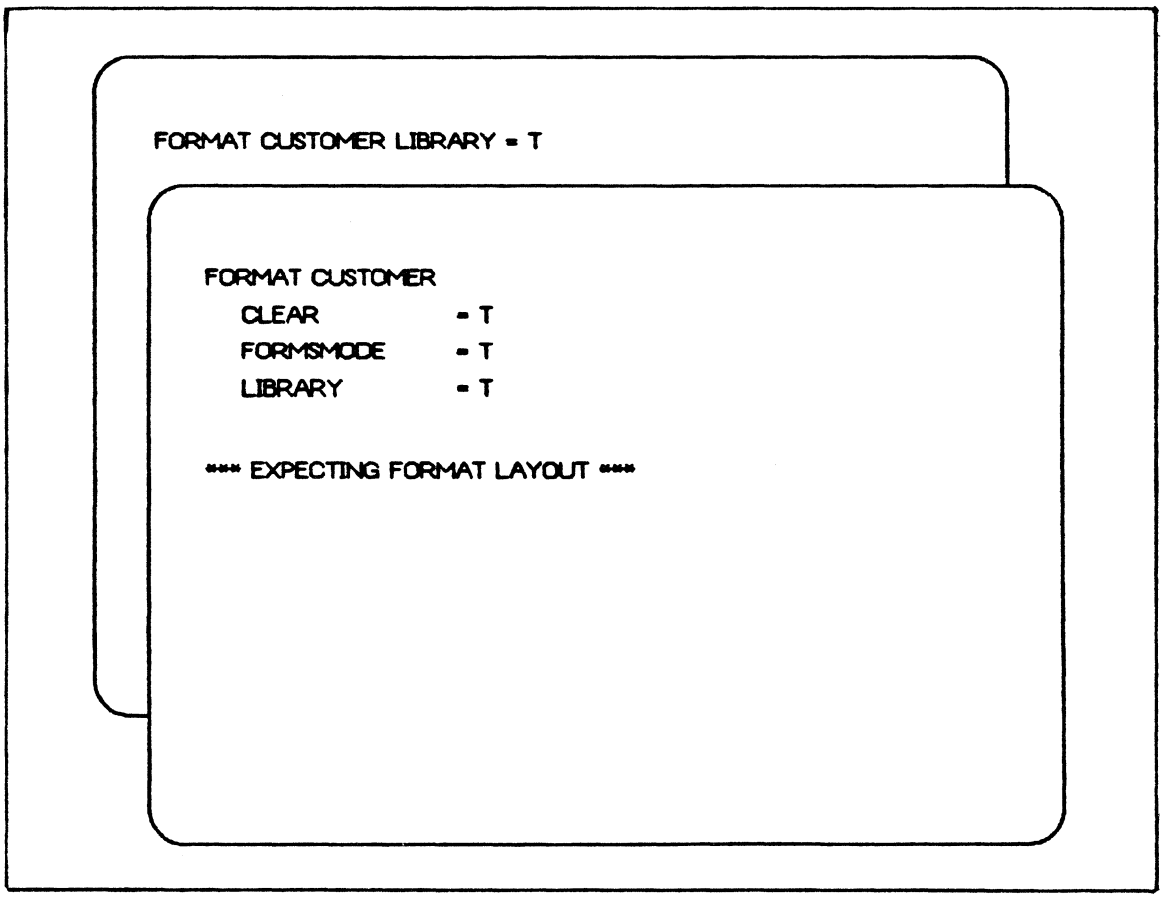


Figure 2-2. Creating a Format

At this point, the terminal must be cleared and the appropriate layout placed on the screen. Figure 2-3 shows the rough layout of the Custmr format both before translation and after the Format Maker response. Each pound sign (#) represents the screen position at which an alphanumeric character from a message sent to the terminal will be displayed. The pound sign characters are not shown on the screen when the format is in actual use.

Laying out a format consists of entering all headings, field identification titles, and field delimiters on the screen. The only special character that may be entered is the pound sign (#), which indicates where alphanumeric data is to be displayed. This data is sent by an application program and will be merged into the format wherever the #'s occur. A maximum of 144 output fields may be entered for one format. An output field is a continuous string of pound signs. For example, "###-##-####" consists of three fields.

As soon as the format layout is transmitted from home position or from some point after the last character of the format layout (transmission from anywhere else causes the format to be truncated), a format is created in the form of TD830 cursor-positioning strings. It is then ready for modification, testing, or transfer to the Live Format file.

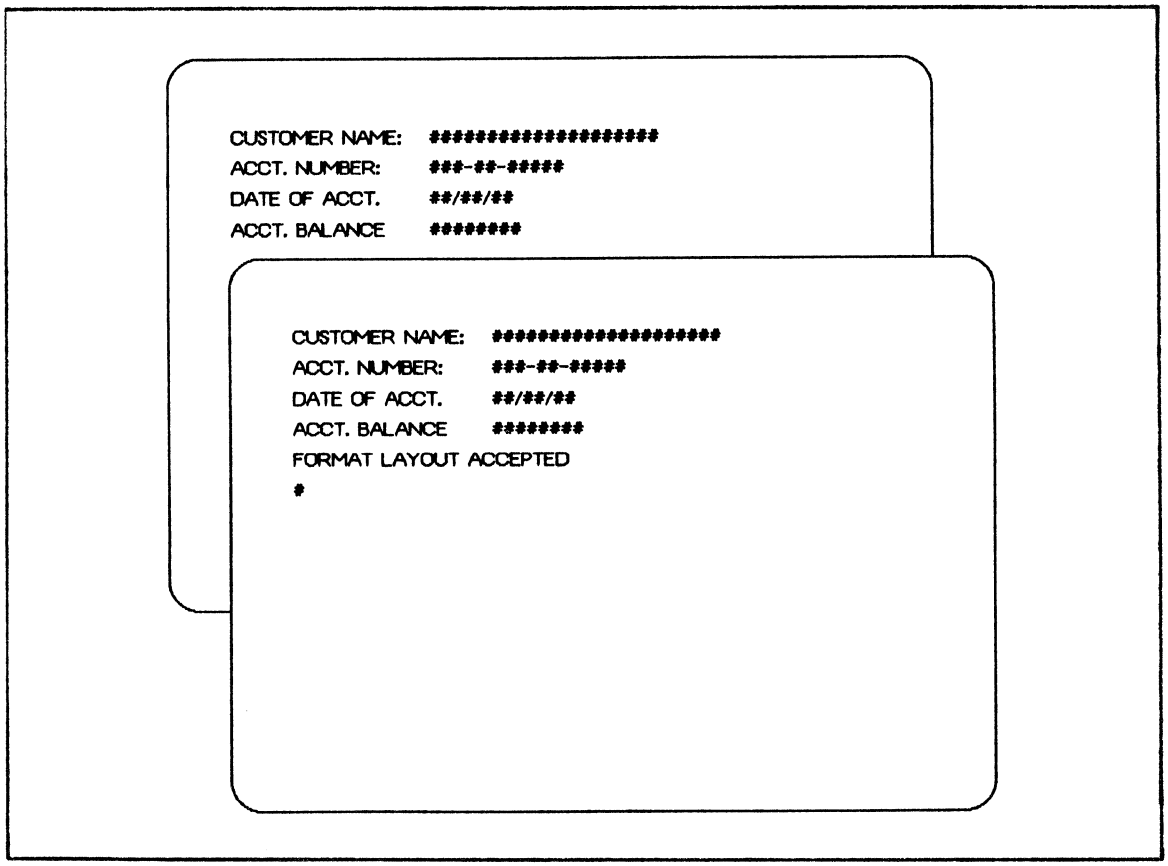


Figure 2-3. Creating a Format Layout

While laying out a format, it may be desirable to end Expecting Format Layout mode to modify the format attributes or to display an existing format. This can be done by issuing the IGNORE command in the home position and may be done at any time prior to transmitting the format layout.

Note that transmission of the IGNORE command at any place but home position will cause that portion of the displayed format that precedes the cursor on the screen to be entered in the Test Format file. Figure 2-4 illustrates the user's entry of IGNORE and Format Maker's response to the cancellation of a format.

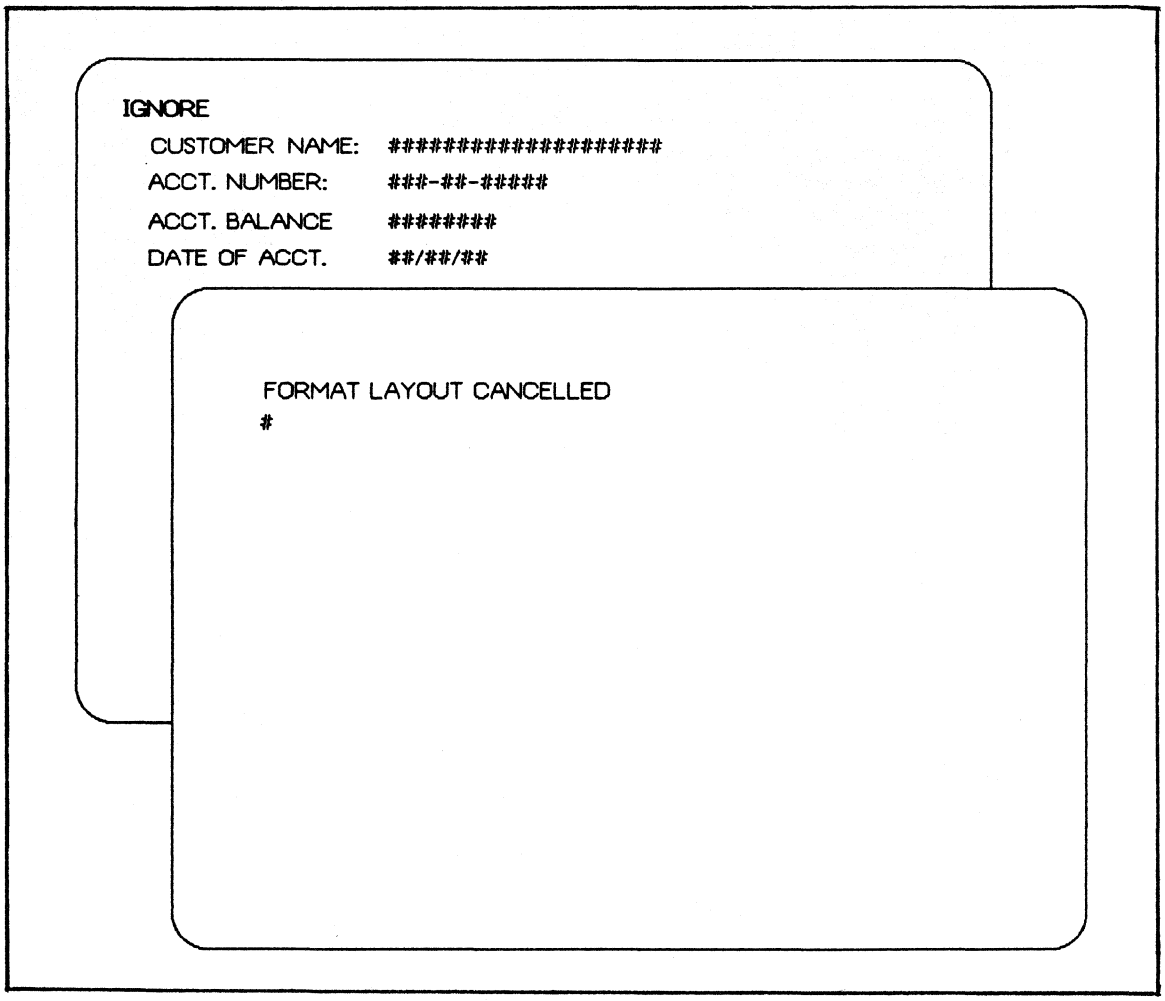


Figure 2-4. Cancelling a Format

MODIFYING A FORMAT

The MODIFY command is used to make changes in an existing format. The first screen of Figure 2-5 shows the user's entry of the MODIFY command to display the Custmr format. The second screen shows Format Maker's response. The third and fourth screens show the user's change and and Format Maker's acceptance of the of the altered format.

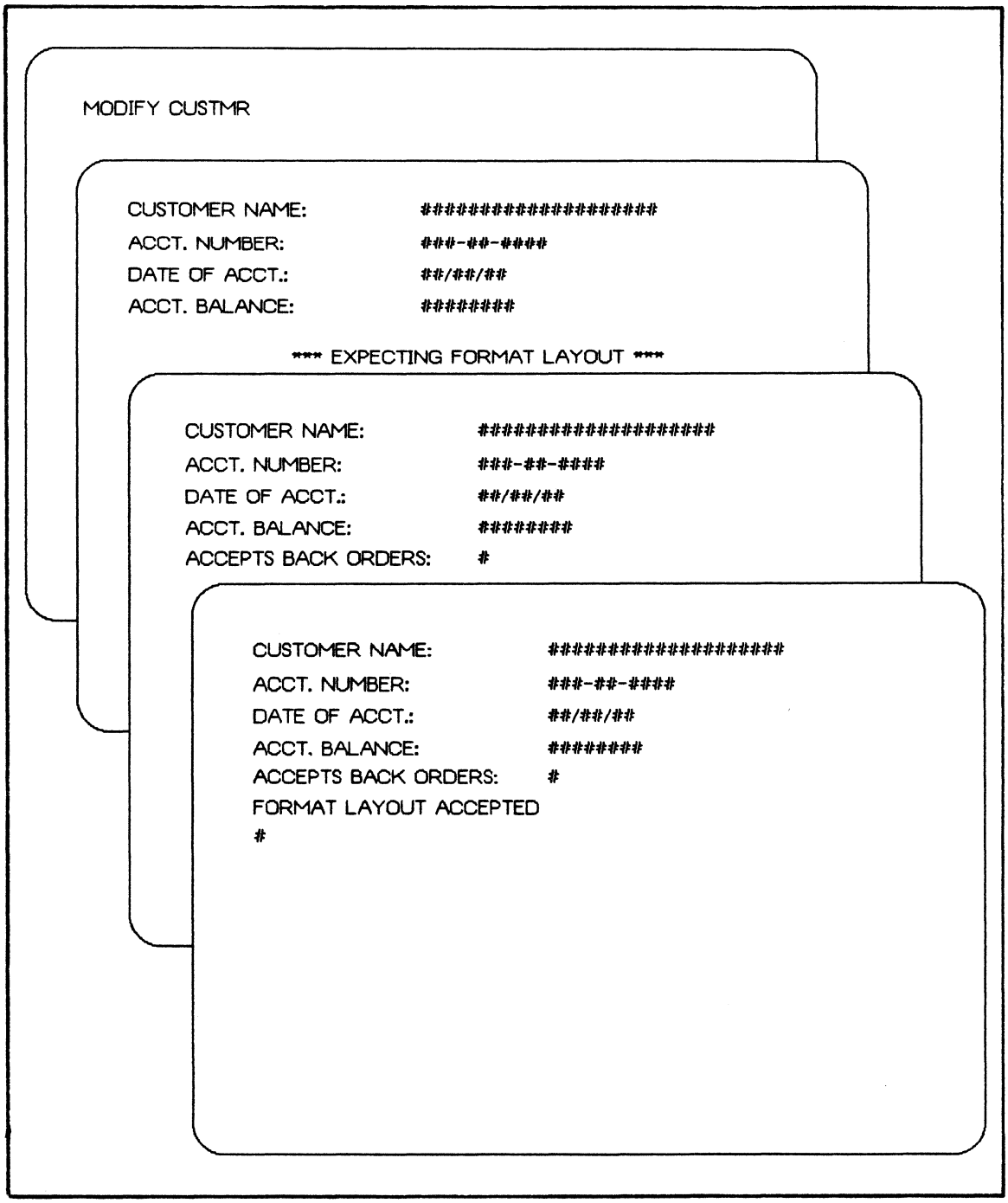


Figure 2-5. Modifying a Format

Unlike the Format Generator program (which is discussed in Section 4), Format Maker has no LINE command to enter control characters which cannot be entered graphically (alphanumerically) on the screen. Instead, any control characters for reverse video, bright highlight, and underlining must be entered using the CTRL H sequences described in the TD830 manual.

The end of a highlighted string is indicated by the right delimiter character that is defined at the beginning of the formatting session. If no right delimiter is used, the remainder of the line is highlighted.

TCL formatting features that cannot be represented in a format layout (data translation, repetition of fields, and location specifiers) are not available in the Format Maker. However, these features may be be used in the Format Generator program (see Section 4).

TESTING A FORMAT

As soon as a format layout has been transmitted, it is ready to be tested using the TEST command. This can be done in one of two ways. The first way is to enter the TEST command with no message attached. The format will be displayed in forms mode as it will appear during actual use. No data will be shown in the fields of the format.

The second way is to enter the TEST command with a single alphanumeric string attached (exact syntax may be found in Section 3). The format will be presented as in the first case, but data from the test string will be displayed. Figure 2-6 illustrates the testing of the Custmr format. The first screen shows the user's entry of the TEST command with data. The second screen displays the format with data applied as it appears in a "live" environment.

To exit the forms mode, enter CTRL Q.

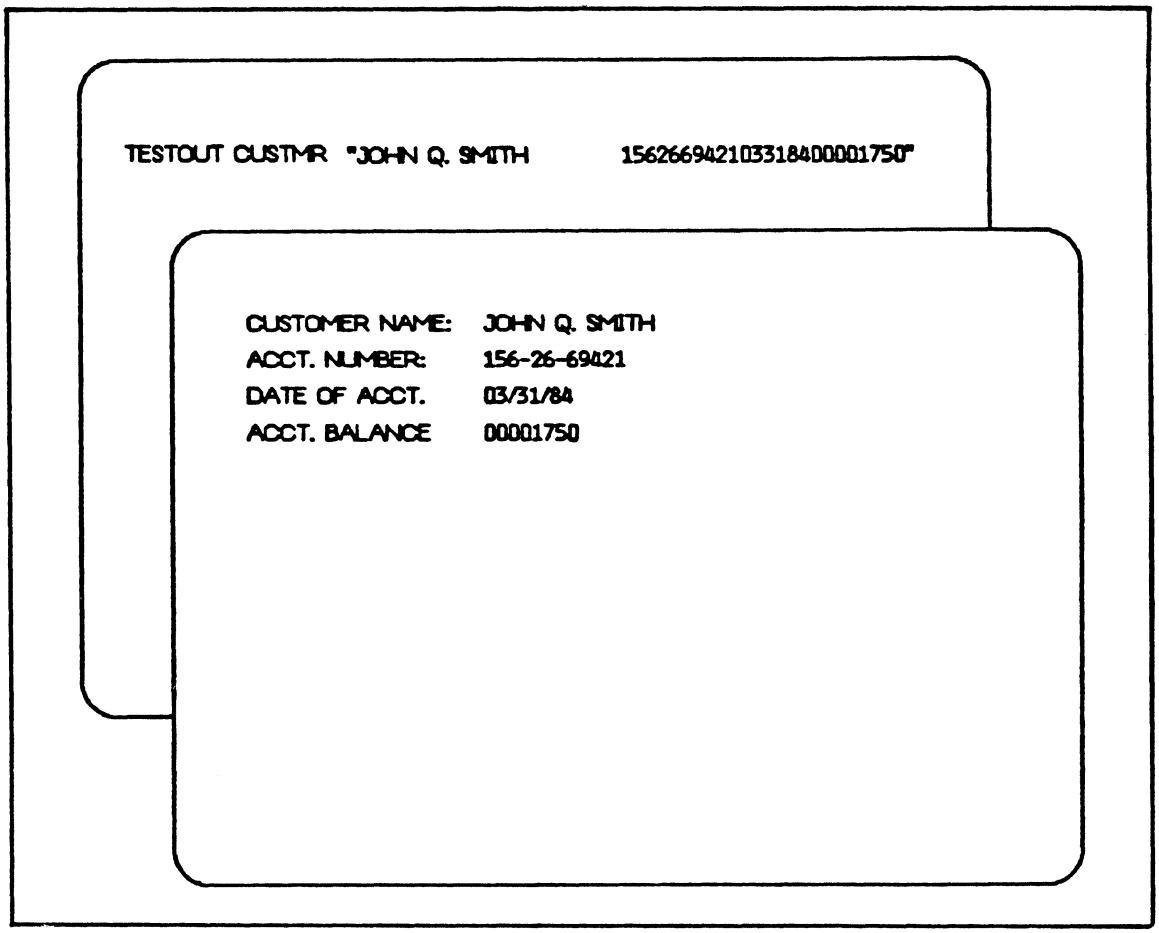


Figure 2-6. Testing a Format

RETRIEVING A FORMAT

The GET command duplicates an existing format from another format file for modification or transfer into a live environment. The format must already exist in the specified file. If no file is specified, an error message is returned. See "Creating a Live Format File" in this section for information on creating other format files.

Upon successful completion of a GET command, the terminal displays a picture of the retrieved format with any previous modifications intact. The GET command does not insert the retrieved format in the Test Format file. However, by using the two-page capability of the TD830 terminal, a user can GET a format layout, switch pages to execute the FORMAT command, and switch back to transmit the source images as a new format. The result is two identical formats with

different names. The duplicate format can be modified as desired.

The first screen of Figure 2-7 shows the user requesting the Custmr format from the MCSFORMATS file. The second screen shows Format Maker's response.

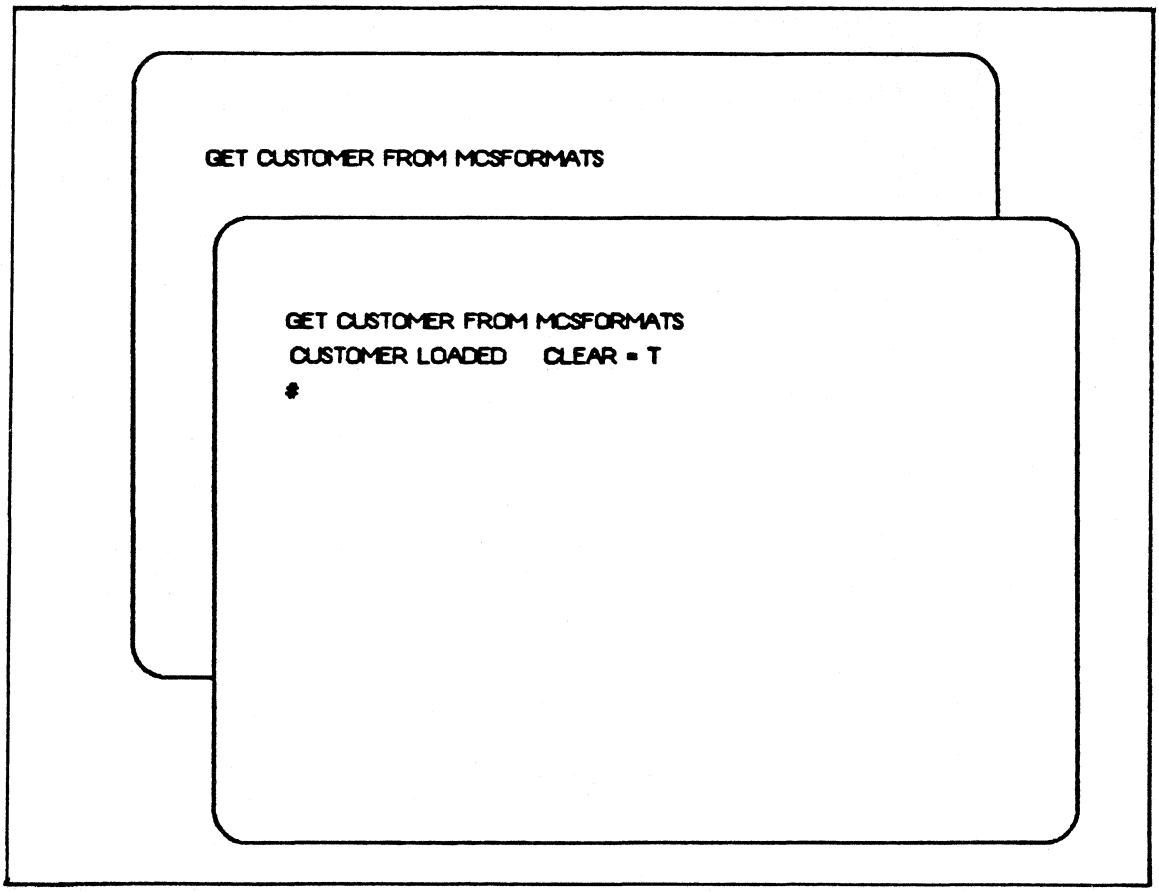


Figure 2-7. Retrieving a Format

DISPLAYING A FORMAT

Two commands can be used to examine the current format layout. The first command, DISPLAY, sends the format layout to the terminal. If no format name is specified, the most recently accessed format is displayed. The second command, WRITE, sends the format layout to the line printer. Both commands are useful when a temporary or permanent display of the format is required.

Figure 2-8 demonstrates the DISPLAY command. In this example, it is assumed that the Test Format file contains the Custmr format, as used in previous examples.

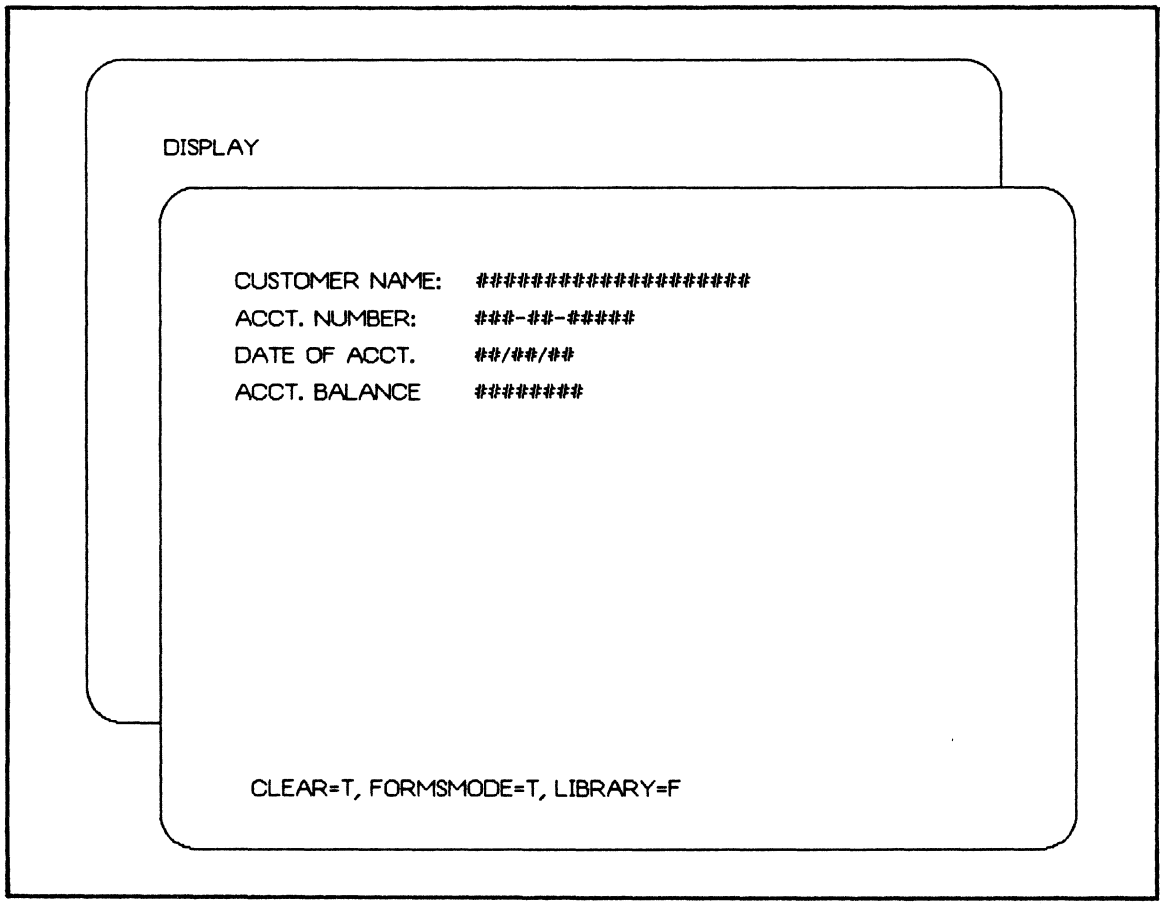


Figure 2-8. Displaying a Format

UPDATING A FORMAT

It is often desirable to modify a format in a live GEMCOS environment. This can be done by using the GET command to display a format from the Live Format file. Once the format layout has been modified, it must be copied back to the Live Format file. This can be done by placing the modified format on the second page of the terminal display and entering the FORMAT command. With the terminal in Expecting Format Layout mode, the modified format can be displayed and transmitted. Once the new format has been tested, the UPDATE command transfers it to the Live Format file.

The updated format, however, is not recognized by the MCS until the *UPD FORMATS command is entered from a GEMCOS control station. A maximum of 400 format updates can be performed before it is necessary to enter the *UPD FORMATS command.

Example:

It is necessary to modify one of the lines of the Custmr format in the GEMCOS format file GEMCOS/NONINTERPS. The steps involved are as follows:

1. Display a format from the Live Format file by entering:

   GET CUSTMR FROM GEMCOS/NONINTERPS

2. Modify the format as needed and shift to the second page.

3. Introduce the modified format into the Test Format file by entering the following:

   FORMAT CUSTMR

   Then bring the format to the front page for transmission.

4. Transfer the format to the Live Format file using the UPDATE command:

   UPDATE TO GEMCOS/NONINTERPS CUSTMR

5. Modify the header records of the Live Format file to allow the GEMCOS MCS access to the updated formats by entering the following at a GEMCOS control station:

   *UPD FORMATS

   To similarly modify any other format file, change the file name to GEMCOS/NONINTERPS then enter *UPD FORMATS from a GEMCOS control station.

CREATING A LIVE FORMAT FILE

When Format Maker is initiated, it creates a test file with a record size of 1620 bytes and assigns it the name MCSNONFORM/<hhmm>, where <hhmm> is the current time represented according to the standard 24 hour clock. This file is identical to the Live Format file used by the GEMCOS MCS. If no Live Format file currently exists, one can be created by changing the name MCSNONFORM/<hhmm> to GEMCOS/NONINTERPS.

If it is necessary to add to or otherwise change an existing noninterpretive format file, the OPEN command should be used. This command closes the current test file and opens the specified file. Nothing need be done to the current file before opening a new file. If the current file has no formats, it will be purged.

SECTION 3

FORMAT MAKER COMMANDS

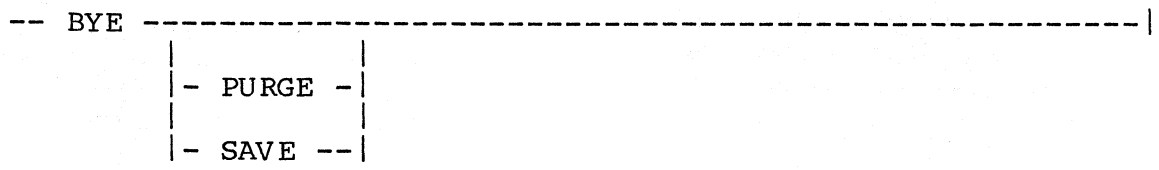
This section provides a complete description of Format Maker commands. "Railroad" syntax diagrams are used to describe the syntax of these commands. An explanation of syntax conventions can be found in the B 1000 Series GEMCOS Reference Manual.

The following is a list of all the Format Maker commands.

BYE
DEBUG
DELETE
DISPLAY
FORMAT
FORMAT LAYOUT
GET
HELP
IGNORE
LIST
MODIFY
NULL INPUT
OPEN
STATUS
TEACH
TEST
UPDATE
WHAT
WRITE

BYE

Syntax:



Semantics:

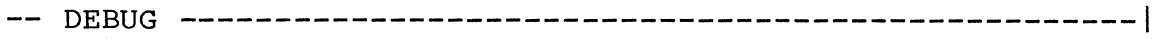
The BYE command terminates the user's current formatting session. If BYE PURGE is entered, the test file is purged. If BYE SAVE is entered, the test file is saved. If no option is specified and the Test Format file contains no formats, the file is purged. If no option is specified and the Test Format file does contain formats, the file is saved.

Example:

```
BYE SAVE
```

DEBUG

Syntax:



Semantics:

The DEBUG command is used as a "toggle" which alternately sets and resets debugging logic built into the Format Maker. The DEBUG command does not affect the development of formats but is designed to help diagnose difficulties within actual Format Maker code.

When the debugging logic is invoked, the following information is written to a printer file:

- A monitor of all procedures entered, including pertinent data for that procedure.
- A program dump. This occurs whenever the Format Maker encounters a serious or fatal error.

Example:

```
DEBUG
```

DELETE

Syntax

```
-- DELETE ---------<format>--------------------------------|
```

Semantics:

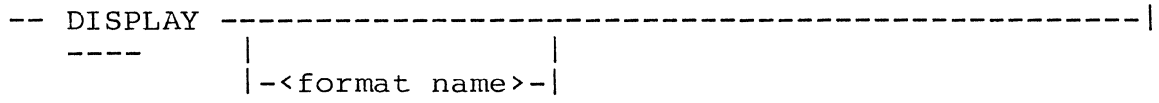
The DELETE command is used to delete formats from the Test Format file. If, after deletion, the STATUS or WHAT command is used, no current format will be listed.

Example:

```
DEL OE23
```

DISPLAY

Syntax:



Semantics:

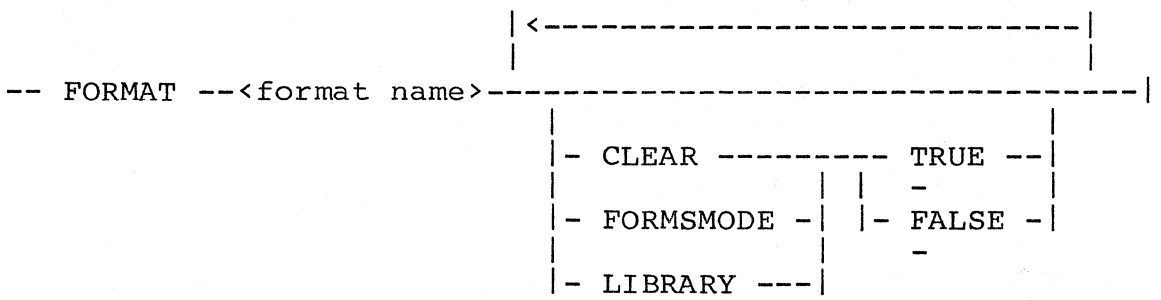
The DISPLAY command is used to display the current format or the specified format on the terminal. If no format name is specified, the current format is displayed. If there is no current format, an error message is returned. The format's attributes are displayed on the status line (line 25).

Example:

```
DISPLAY FORM51
```

FORMAT

Syntax:



Semantics:

The FORMAT command is used to create a blank format, assign attributes to that format, and place the terminal in Expecting Format Layout mode. A format name can be a maximum of six characters in length. The format attributes have the following meanings:

- CLEAR inserts the contents of the CLEAR define at the beginning of the format. The default value of the CLEAR define is 4"0C". The default value for the CLEAR attribute is TRUE.

- FORMSMODE inserts the contents of the FORMSMODE define at the end of the format. The default value of the FORMSMODE define is 4"27E6". The default value of the FORMSMODE attribute is true. Any allowable characters are acceptable as forms characters. The symbols US, GS, and FS may be used to start a data field, and the RS symbol may be used to end a data field.

- LIBRARY causes the Format Maker to produce a COBOL library corresponding to the output fields described in the format. The COBOL library will have a "01" level with the format name followed by "-RECORD-". The individual fields (05 levels) will be of the form:

  <format name>-<line #>-<field # on line>.

Unless there is an attribute error found in the input, the FORMAT command always leaves the terminal in an Expecting Format Layout mode. This means that a format has been named and Format Maker is now expecting a layout of the format in the next transmission. Once the terminal is in Expecting Format Layout mode, the user must do one of three things:

- Clear the screen, layout a format on the screen, and transmit the format layout.
- Enter IGNORE in home position, which removes the terminal from Expecting Format Layout mode. Then enter another command, or start over with a new format name and attributes.
- Enter DISPLAY, which displays a previously created format layout for correction and leaves the terminal in an Expecting Format Layout mode.

To determine if the terminal is in an Expecting Format Layout mode, a STATUS or WHAT command can be entered without changing or affecting the current format or terminal status.

Examples:

```
FORMAT FMT1
FORMAT SF09  CLEAR = F   FORMSMODE=TRUE
```

FORMAT LAYOUT

The format layout is not a command. It is simply the format laid out by the user and sent to the Test Format file (MCSNONFORM) when the terminal is in Expecting Format Layout mode.

See Section 2 for an explanation of how to lay out a format.

GET

Syntax:

```
-- GET -----<format name>---- FROM <file name> ------------|
```

Semantics:

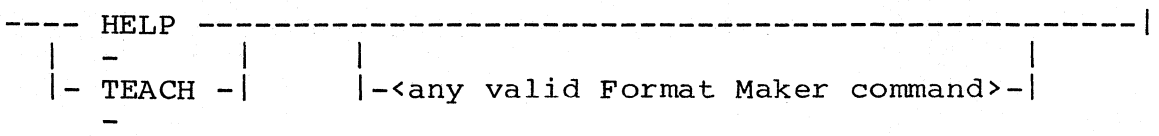
The GET command is used to retrieve a format from a noninterpretive format file. The format will be displayed on the user's terminal. The format does not become the current format, but is displayed only for viewing purposes.

Example:

```
GET FORM02 FROM FORMFILE/AP ON USERPAC
```

HELP

Syntax:



Semantics:

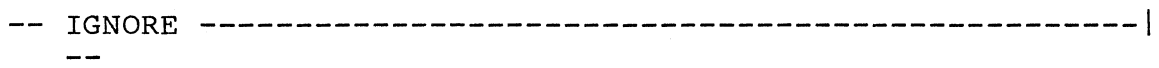
The HELP command is used to obtain a list of valid Format Maker commands or to learn the syntax of a particular command.

Example:

```
HELP
```

IGNORE

Syntax:



Semantics:

The IGNORE command terminates the command that is in progress. The IGNORE command can be used only when the terminal is in an Expecting Format Layout mode. Upon receipt of this command, the terminal is taken out of Expecting Format Layout mode, and another command can be entered. See Section 2 for more detail.

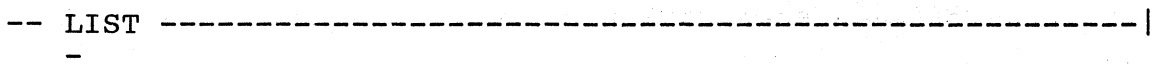
This command must be entered starting at the first position on the terminal, and IGNORE must be the only word entered.

Example:

```
IGNORE
```

LIST

Syntax:



Semantics:

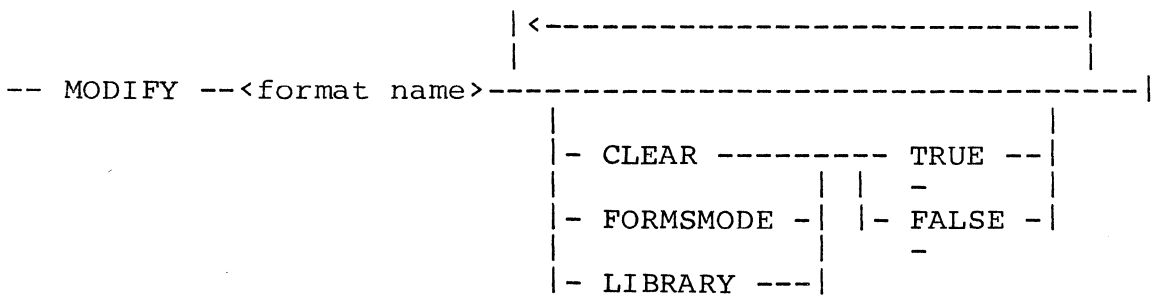
The LIST command displays the names of all formats in the test file on the terminal. Deleted formats are not included in the list. If there is a current format, it will be displayed in reverse video. Should there be more names than can fit on one page, the heading "---MORE---" will be displayed. Transmitting a blank character to the Format Maker causes the next screen of names to be displayed.

Example:

```
LIST
```

MODIFY

Syntax:



Semantics:

The MODIFY command changes an existing format and/or its attributes. Attributes not specified will remain the same. After entry of this command, the terminal will be placed in Expecting Format Layout mode. The user must then enter one of the following:

- The modified format layout.
- IGNORE - to get out of Expecting Format Layout mode.
- DISPLAY - to display the current format.
- STATUS - to see the current status.

Even if only the attributes are to be changed, the format layout still must be transmitted.

Examples:

```
MODOFY OE01
MODIFY FORM1  LIBRARY=T
```

NULL INPUT

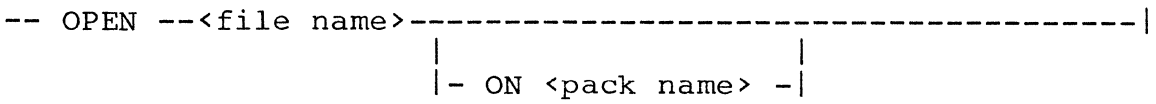
Null input is not a command. It is used to receive the next screen of requested information after a LIST command is entered. A null input consists of a string of one or more blanks transmitted from home position.

Example:

<one space>

OPEN

Syntax:



Semantics:

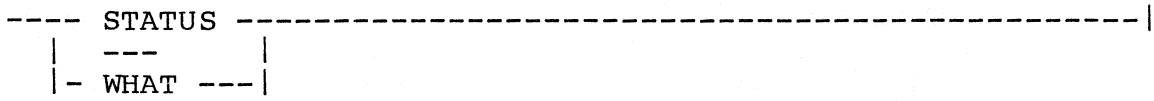
The OPEN command opens another noninterpretive format file and closes the current Test Format file. If the current test file has no formats, it is purged; otherwise it is saved. The file name specified in the OPEN command must be a valid Format Maker format file. This command can be used to add formats to an existing file or modify an existing file.

Example:

```
OPEN MCSTESTFOR/01 24 ON GEMPAC
```

STATUS

Syntax:



Semantics:

The STATUS command is used to obtain information about a particular station that is attached to Format Maker. The information obtained using the STATUS command includes:

- Terminal width and length (always 80 by 24).
- Current DEBUG setting.
- Name of the current format (if any).
- Name of the current Test Format file.
- Number of the formats in the Test Format file.
- Number of format updates that were done in this session.
- Number of formats that were deleted in this session.
- Last command entered.

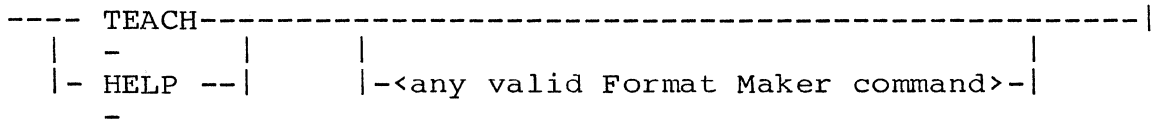
If a STATUS command is entered while the terminal is in an Expecting Format Layout mode, the status report includes the name of the new format to be entered into the Test Format file and displays the message: EXPECTING FORMAT LAYOUT. While in this mode, the STATUS command must be entered starting at the first position on the terminal, and STATUS must be the only word entered. The STATUS and WHAT commands are synonymous.

Example:

```
STATUS
```

TEACH

Syntax:



Semantics:

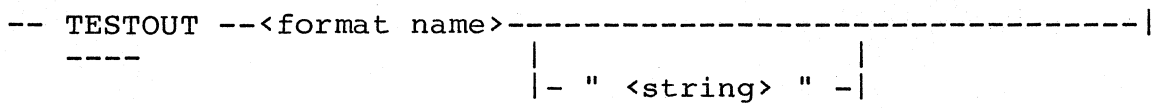
The TEACH command is used to obtain a list of valid Format Maker commands or to learn the syntax of a particular command.

Example:

```
TEACH
```

TEST

Syntax:



Semantics:

The TEST command allows a format to be applied to a test message, and to have the resulting formatted message returned to the terminal. The test message is a character string set off by quotation marks, such as: "12XYZ". Note that, unlike the Format Generator program, Format Maker only allows the entry of a single string, set off by quotation marks.

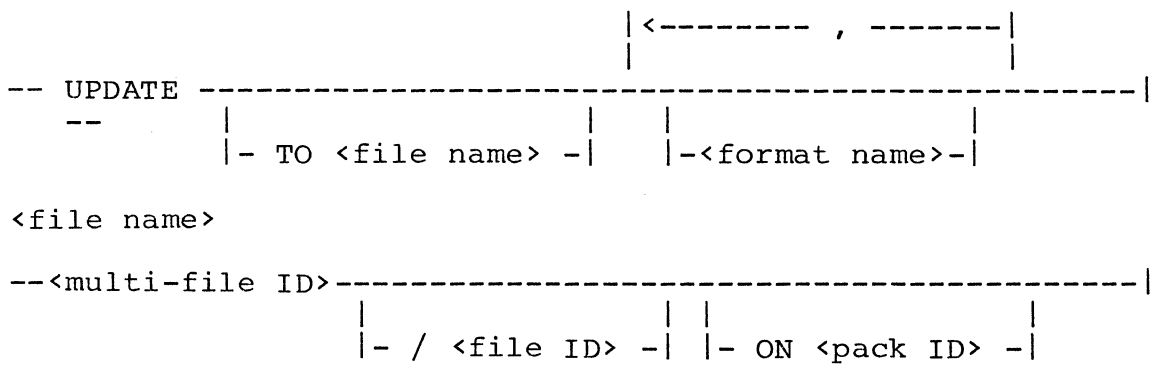
If too much or not enough test data is entered, a warning message is sent to the terminal's status line after the test message is applied to the format. If no test data is entered, the format is applied to a blank message, as in a forms request in a "live" GEMCOS environment. Since Format Maker creates only output formats, the TESTIN command used by Format Generator is not used by Format Maker.

Examples:

```
TEST FA1  "TEST DATA"
TESTOUT OE01
TESTOUT 148 "1234567890"
```

UPDATE

Syntax:



Semantics:

The UPDATE command is used to place one or more test formats into the live noninterpretive format file. The formats may be existing formats or new additions to the other file. This command is only valid if the Format Maker is run in update mode (SW0 = 0).

If no file name is specified, the current live GEMCOS format file named GEMCOS/NONINTERPS is used. Otherwise, the specified file is updated. If no formats are specified, all formats are updated. Otherwise, only the specified formats are updated.

As with interpretive formats, the updated formats do not actually become live until the user enters the following GEMCOS command from a GEMCOS control station.

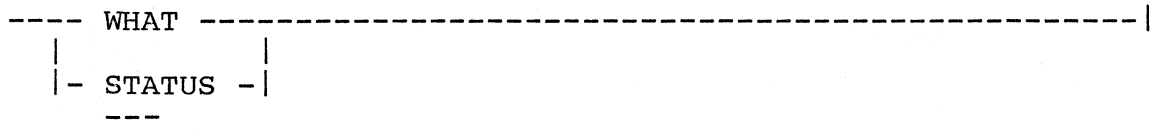
```
*UPD FORMATS
```

Examples:

```
UPDATE
UP TO NONFORMS/PAYROLL ON PACK4  FORM01, F8
UPDATE FAR01, FAR03, FAR06
```

WHAT

Syntax:

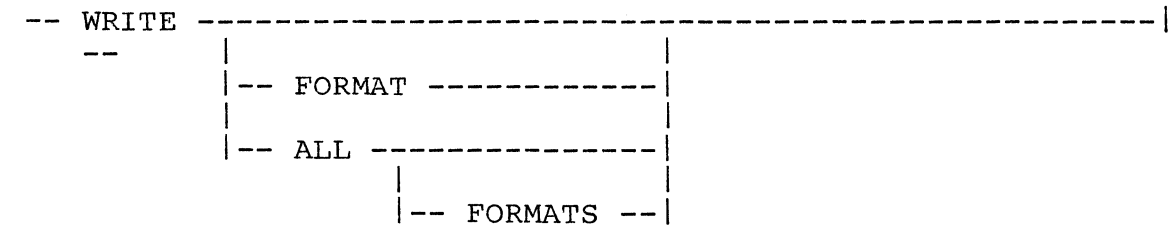


Semantics:

The WHAT command uses the same syntax and returns the same information as the STATUS command.

WRITE

Syntax:



Semantics:

The WRITE command causes the current format or all formats in the test file to be written to the line printer.

Examples:

```
WR
WRITE ALL FORMATS
```

SECTION 4

FORMAT GENERATOR FORMATTING SESSION

The Format Generator program creates input and output formats for use by terminals within a GEMCOS network. Each format is compiled into a set of GEMCOS TCL formatting codes ("op" codes), each 2-digit number of which refers to a separate formatting feature. When a compiled format is called by an application program, it is translated into a form usable by the terminal. Format Generator programs are able to insert, substitute, and rearrange the fields of a message sent through it.

A formatting session with its required commands is described in this section, including the following tasks:

- Initiating the Format Generator program.
- Creating and modifying a format.
- Compiling the format.
- Testing the format.
- Transferring the format to the Live Format file.

The following file manipulations related to a formatting session are also described:

- Modifying the TCL Work File record.
- Merging the TCL Work File records.
- Saving the TCL Work File.

FORMAT GENERATOR FILES

The following files are used during a session:

- TCL Work File.
- Format Layout file.
- Test Format file.
- Defines file.
- Live Format file.

The TCL work file contains the TCL source images for formats and functions. It is built by entering sequenced source statements, or by translating the Format Layout file. The TCL work file is purged at the end of a session. However, the TCL work file can be saved to disk at any time during the session.

The Format Layout file contains a picture of the current format that is broken down into lines based on the width of the terminal. It is modified by creating or accessing a format. The Format Layout file is purged at the end of a session.

The Test Format file contains all the formats and functions created during the current session. It is modified by compiling formats and functions and is purged at the end of a session.

The Defines file contains a list of character defines that can be implemented by the user. A default Defines file is supplied with the Format Generator program, but the user can create a new one using the DEFINE command.

The Live Format file contains those formats currently accessible to the GEMCOS MCS.

INITIATING THE FORMAT GENERATOR PROGRAM

The Format Generator program can be initiated in one of two ways:

- If not running under the control of a GEMCOS MCS, the Format Generator is executed in the same manner that any application program is executed:

  ```
  EX GEMCOS/FORMGEN <optional file equates>
  ```

- If running under the control of a GEMCOS MCS, the GEMCOS EX network control command is entered from the user's station. The following must be true:

  ```
  MAXASSIGNERS = 10.
  INTERFACE    = NONPARTICIPATION.
  ```

Note that the terminal should not be in SCROLL mode when a formatting session is initiated.

After the Format Generator program goes to beginning-of-job (BOJ), it waits to receive input from the station. Upon receipt of any input from the station, the program sends a message indicating the start of a session.

A maximum of 10 stations can use the same copy of the Format Generator program concurrently. The actual number of stations attached to a single copy depends on the number of stations contained in the opened remote file. Multiple copies of the Format Generator can be run simultaneously since all workfiles are uniquely identified.

BEGINNING A FORMATTING SESSION

After the Format Generator program greeting is received, the user can begin formatting with the entry of the first command. Commands are entered in a free-form manner from anywhere on the terminal screen, except when the terminal is in Expecting Format Layout mode. When the terminal is in Expecting Format Layout mode, commands should be entered starting at the first character position on the screen.

The BYE command ends the formatting session and purges the following:

- The current TCL work file.
- The Format Layout file.
- The Test Format file.

See Section 5 for information on command syntax.

CREATING A FORMAT

To create a format, the FORMAT command must be issued. This places the terminal in Expecting Format Layout mode. When in this mode, the user can lay out the format as it should appear, or cancel the FORMAT command. A FORMAT command does not alter the contents of the Format Layout file, but does clear the terminal screen and prepare the Format Generator program to receive the anticipated format layout. Figure

4-1 illustrates the user's entry of the FORMAT command and the response by the Format Generator. In Figure 4-1, the CLEAR and FORMSMODE attributes were selected.

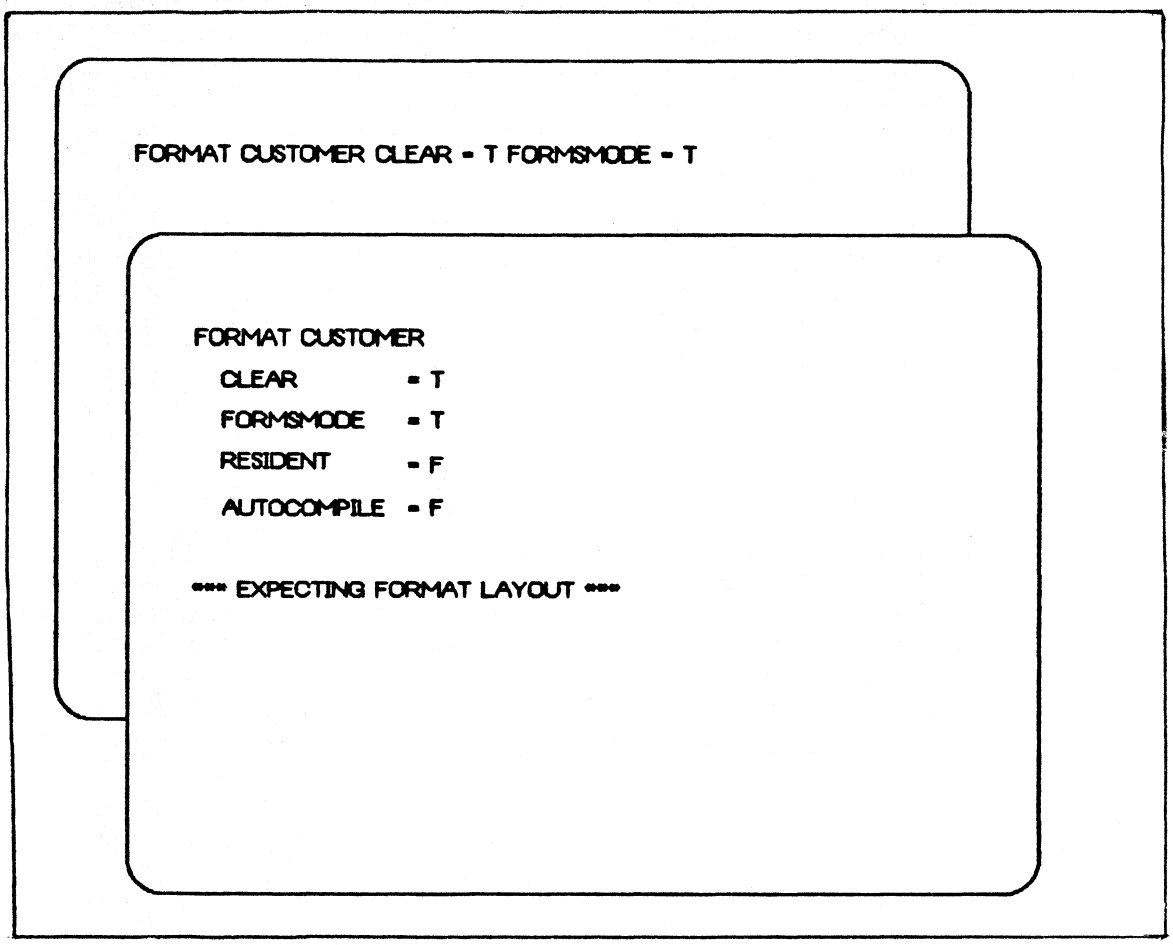


Figure 4-1. Creating a Format

At this point, the terminal must be cleared and the appropriate layout must be placed on the screen.

Laying out a format consists of entering all headings and field indentification titles, positioning forms characters, etc., on the screen. Pound signs (#) and ampersands (&) are used to indicate where data is to be displayed or entered.

A pound sign (#) indicates where one character of alphanumeric data should be displayed, and an ampersand (&) indicates where a character of numeric data should be displayed. The carriage return character (ii) indicates the end of significant data on a line. The user can insert this character on any line to indicate the end of information on that line. In the absence of this character, Format Generator automatically determines the end of significant data for that line.

The format is created upon transmission and becomes available for subsequent processing. The Format Layout file now reflects the format layout just received. Any information previously stored in the Format Layout file is removed.

Figure 4-2 shows the rough layout of the Customer format (which will be used for illustration throughout this section), both before transmission and after the Format Generator response.

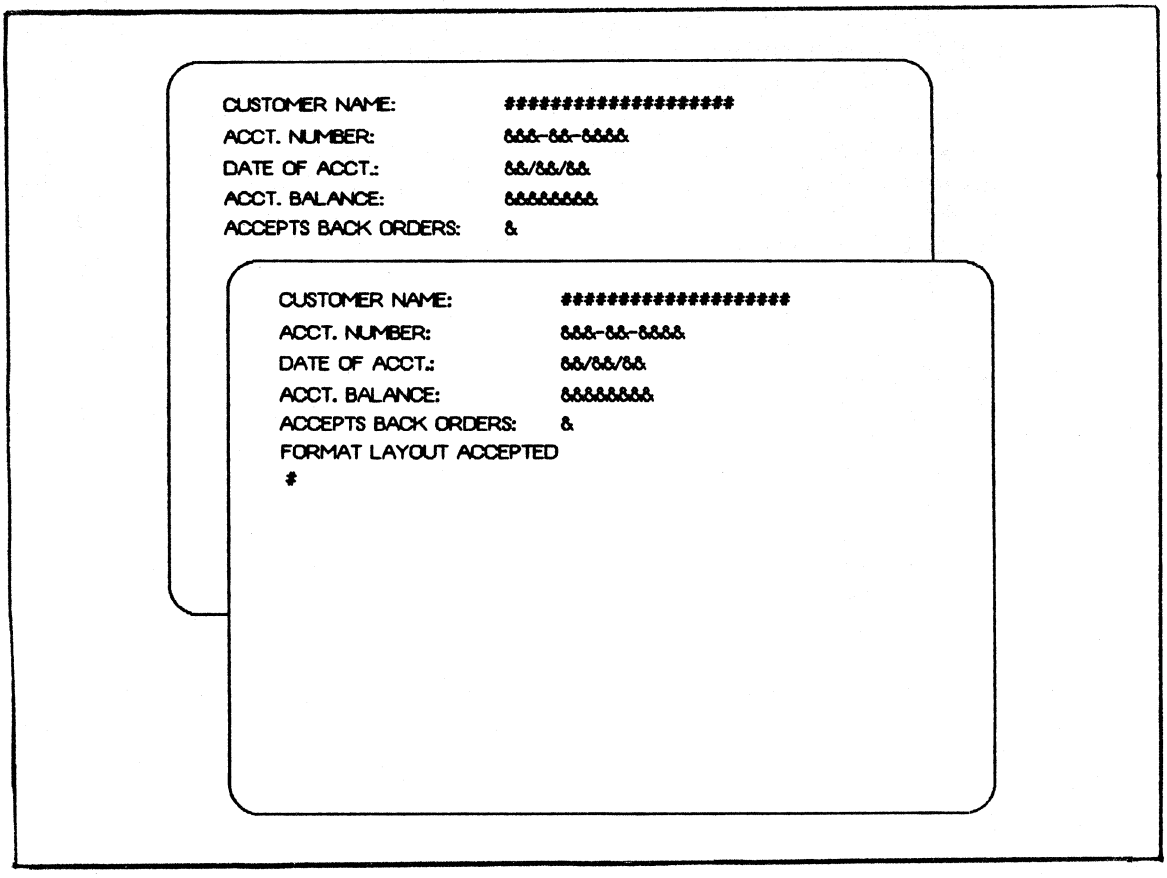


Figure 4-2. Creating a Format Layout

While creating a format, it may be desirable to end Expecting Format Layout mode to carry out some other operation. This can be done by issuing the IGNORE command any time prior to transmitting of the format layout. Figure 4-3 illustrates the user's entry of IGNORE and the Format Generator's response to the cancellation of a format.

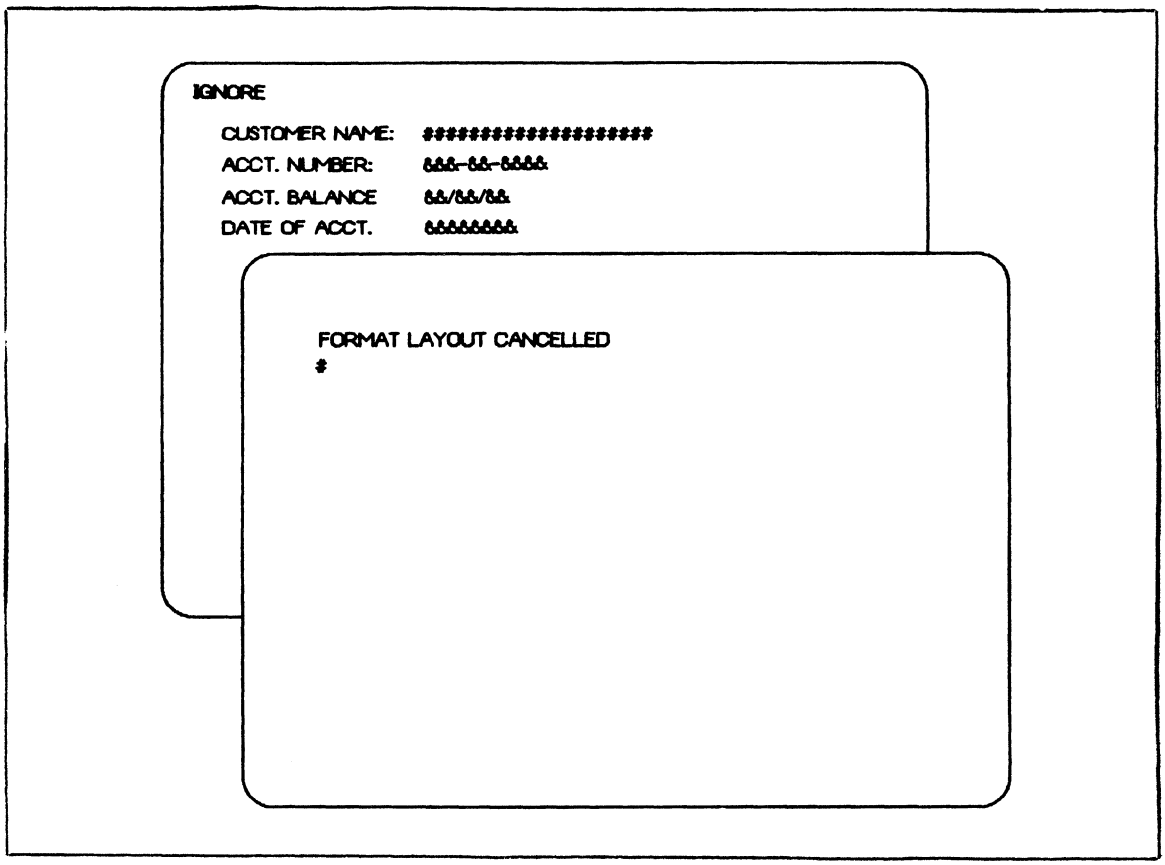


Figure 4-3. Cancelling a Format

MODIFYING A FORMAT

It is not always possible to lay out in Expecting Format Layout mode everything required in a format. Certain terminal control characters cannot be entered graphically (alphanumerically) on the screen. In addition, there are certain TCL formatting features, such as translation, repeats, and location specifiers, that cannot be represented in a format layout. These format additions are inserted into the format using the LINE command.

CONTROL CHARACTER INSERTION

Figure 4-4 illustrates how to insert control characters using the LINE command. The first screen shows the user's first entry, a request to display the current information on line 2. The second screen is the Format Generator's response. The second line of this screen indicates all

literals on that line. The third line indicates the data characters in the message. The third screen shows information added by the user in order to insert the required control characters. The fourth line of this screen indicates where control strings are to be added. The fifth line displays the control strings used. For a break-down of this exchange, see the following table.

Screen	Line	Explanation
Screen 1	Line 1	The user's request to display current information from line 2 of the format.
Screen 2	Line 2	Format Generator returns the literal characters from line 2 of the format.
	Line 3	Data characters in the message of the format.
Screen 3	Line 4	Pointers added by the user to insert control strings. The LINE causes a control string to be inserted immediately before a left angle bracket (<) or immediately after a right angle bracket (>).
	Line 5	4"3F" is an example of a control string, inserted before the left angle bracket. A semicolon indicates the end of each control string (optional for the last control string specified). 4"27","K" is an example of another control string, inserted after the right angle bracket.
Screen 4	Line 6	Format Generator's response to the control character insertion.

The fourth screen is the Format Generator's response to the user's control character insertion.

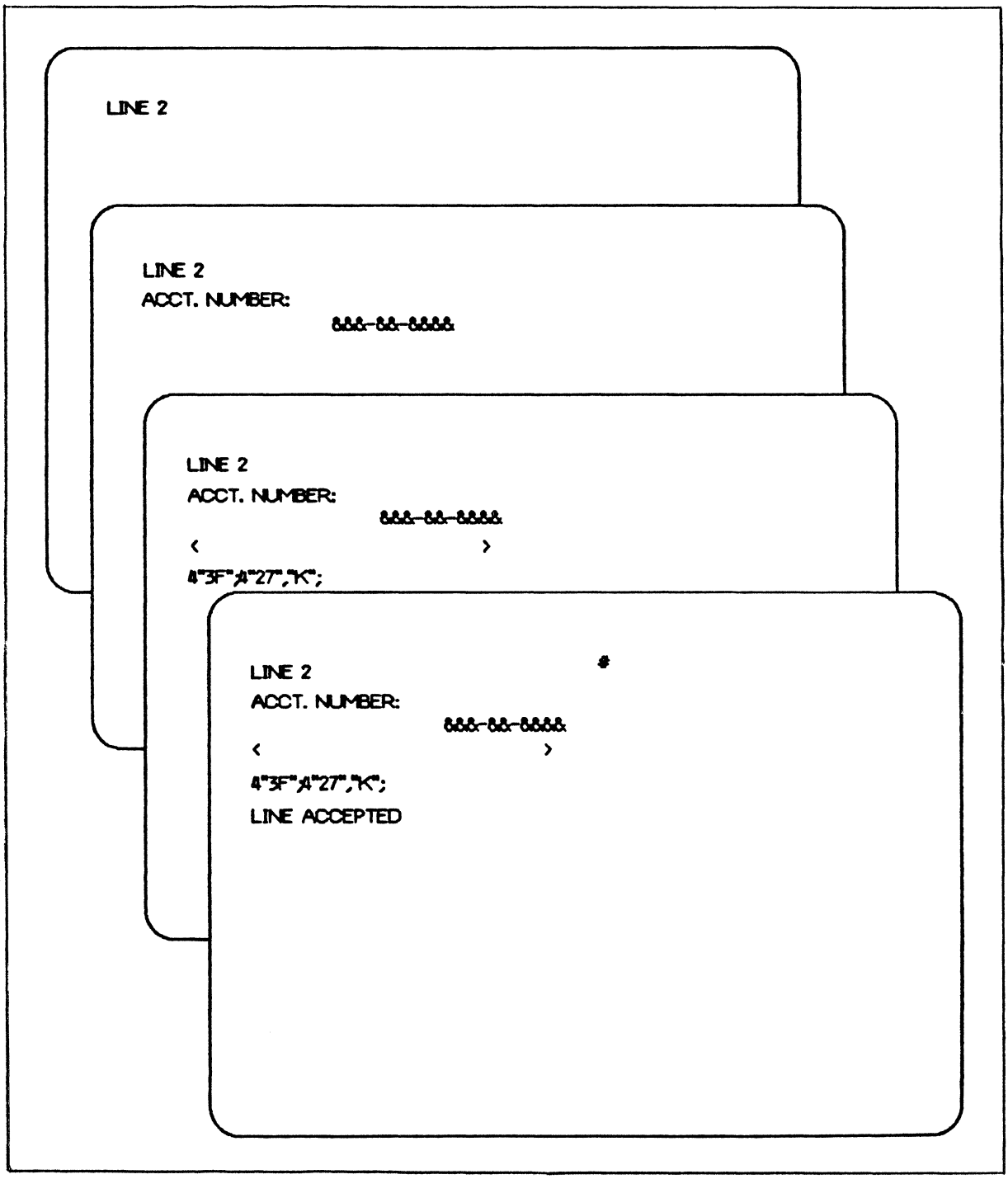


Figure 4-4. Inserting Control Characters

LOCATION SPECIFIERS (REMAPPING)

When the desired field order on a terminal is different from the order declared by the program, or when fields are skipped, message remapping is required. When remapping, it is often helpful to write out the program message layout as in the following COBOL example:

```
01  CUSTOMER-RECORD.
    05  CUST-NAME                  PIC X(20).
    05  CUST-ACCOUNT-NUM.
        10  ACCT-REGION-NUM        PIC 9(3).
        10  ACCT-BRANCH-NUM        PIC 9(2).
        10  ACCT-NUM               PIC 9(6).
    05  CUST-ACCOUNT-DATE.
        10  CUST-ACCT-YEAR         PIC 9(2).
        10  CUST-ACCT-MONTH        PIC 9(2).
        10  CUST-ACCT-DAY          PIC 9(2).
    05  CUST-ACCOUNT-BALANCE       PIC 9(8).
```

Two memory areas are maintained for every message processed through the formatting mechanism. The first contains the message as it appears on the terminal (the external message). The second contains the message as it appears in the program record area (the internal message).

Each area has a pointer associated with it. These are the external message area pointer (external pointer) and the internal message area pointer (the internal pointer). As a field or literal is processed, both pointers are advanced as characters are moved from one area to the other. Both pointers are initially set to zero. As each character in a message is processed, both pointers are advanced by one position.

Data can be rearranged by moving the internal pointer from one field to another. To adjust the setting of the internal pointer, a location specifier ("at" sign (@)) is used with a signed or unsigned integer. When an unsigned integer is used, the internal pointer is adjusted to the absolute position indicated by the integer (one relative). When a signed integer is used, the internal pointer is adjusted in the direction of the sign relative to its present position.

The following are two examples:

@3 Moves the internal pointer to the third character position in the message (absolute remapping).

@-16 Moves the internal pointer 16 characters back (relative remapping).

When remapping is required, the first step involves creating a regular format layout with the fields as they are to appear on the terminal. The LINE command must then be used to identify where the internal pointer must be adjusted to accomplish remapping.

Figure 4-5 shows how to alter the DATE OF ACCT fields so that CUST-ACCT-DAY and CUST-ACCT-DAY are displayed before CUST-ACCT-YEAR on the terminal. The location pointer is first moved forward two positions to access CUST-ACCT-MONTH and CUST-ACCT-DAY, moved back six positions to extract CUST-ACCT-YEAR, and then moved forward four positions for the next field.

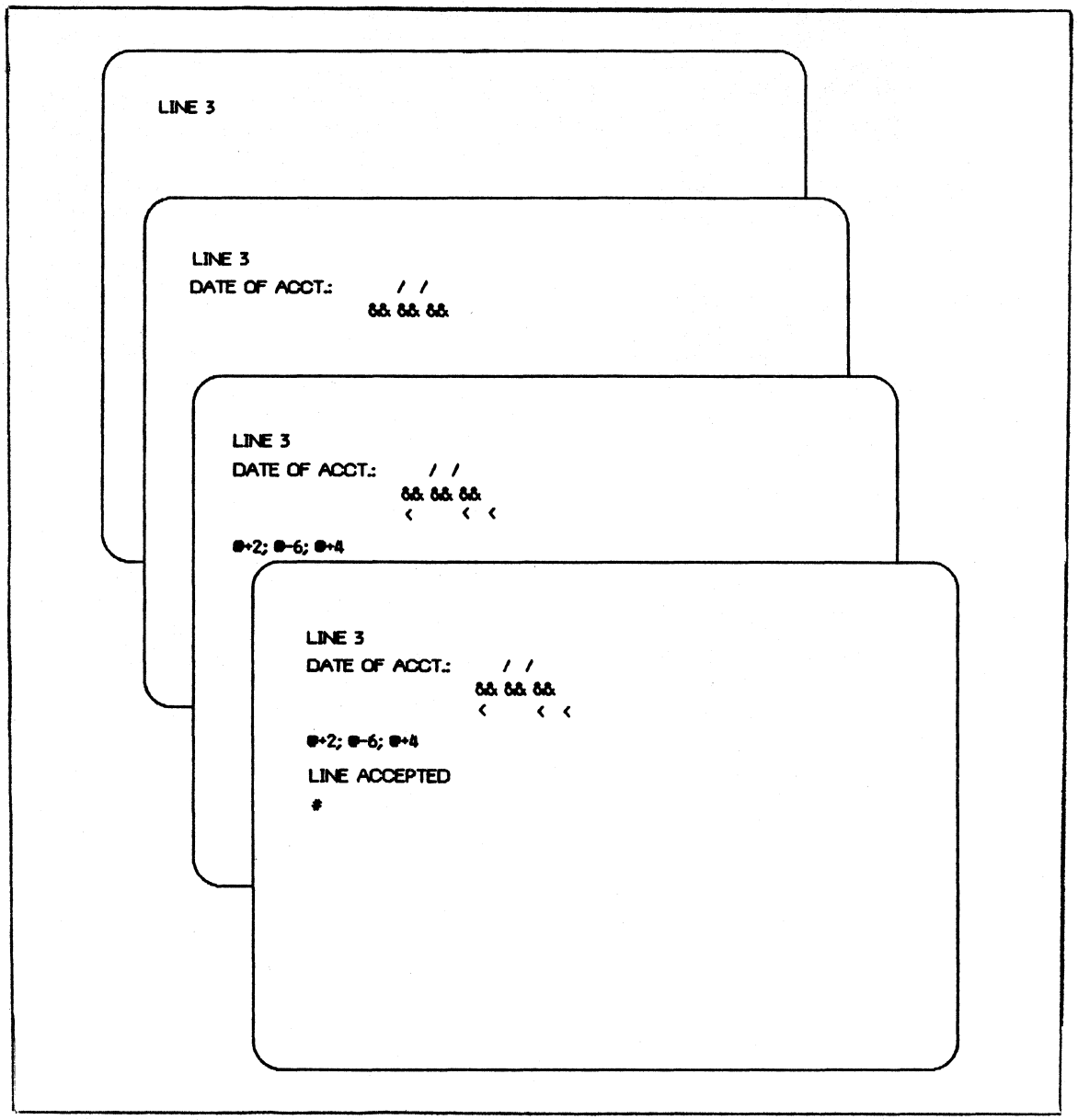


Figure 4-5. Relative Remapping

Figure 4-6 accomplishes the same end as Figure 4-5 functionally, but absolute remapping is used instead of relative remapping. The location pointer is first moved to position 34 to access the month and day, then moved back to position 32 to extract the year, then moved to position 38 for the next field. All character positions are one relative.

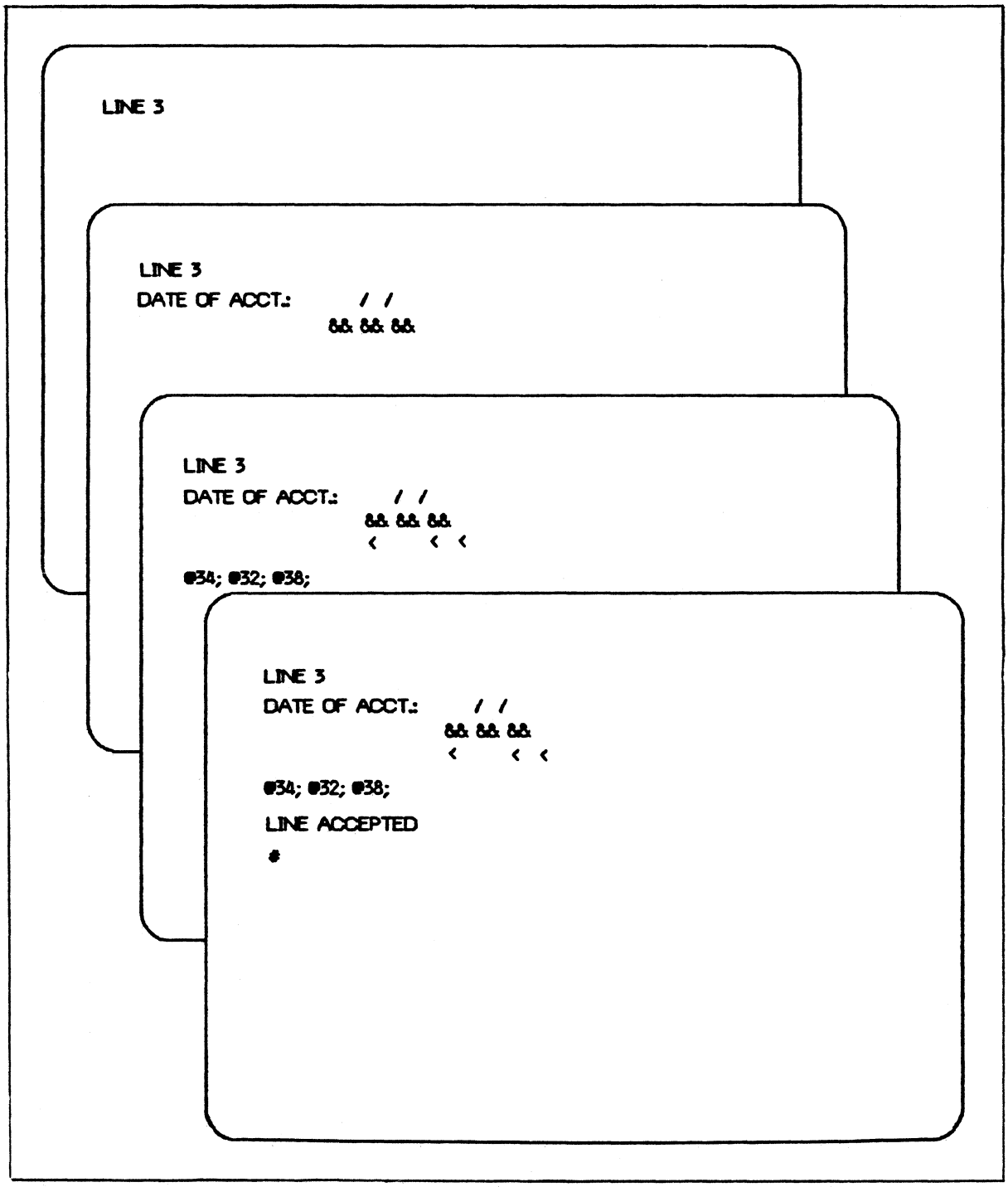


Figure 4-6. Absolute Remapping

TRANSLATION TABLES - TCL FUNCTIONS

A function is a translation table which can be referred to by a format. Refer to the B 1000 GEMCOS Reference Manual for a discussion of functions. To use a function, it must first be declared, using the SEQUENCE command, and compiled, using the COMPILE command. Refer to "Compiling a Format" and "Modifying the TCL Work File" in this section for an explanation of this process. See Section 5 for command syntax.

Functions are inserted into a format using the LINE command, as in the previous examples. In Figure 4-7, the program sends the ACCEPTS BACKORDERS field as a one-byte integer, either 0 or 1. However, it is desired to display the words NO or YES instead. Assuming the function BOOLEAN was previously compiled, this line insertion accomplishes the required translation.

When functions are inserted in this manner, it is necessary to allow room for inserting the characters of the function. All information to the right of the location specifier is shifted to the right by the length of the function. In Figure 4-7, at least six spaces are required at the end of the line.

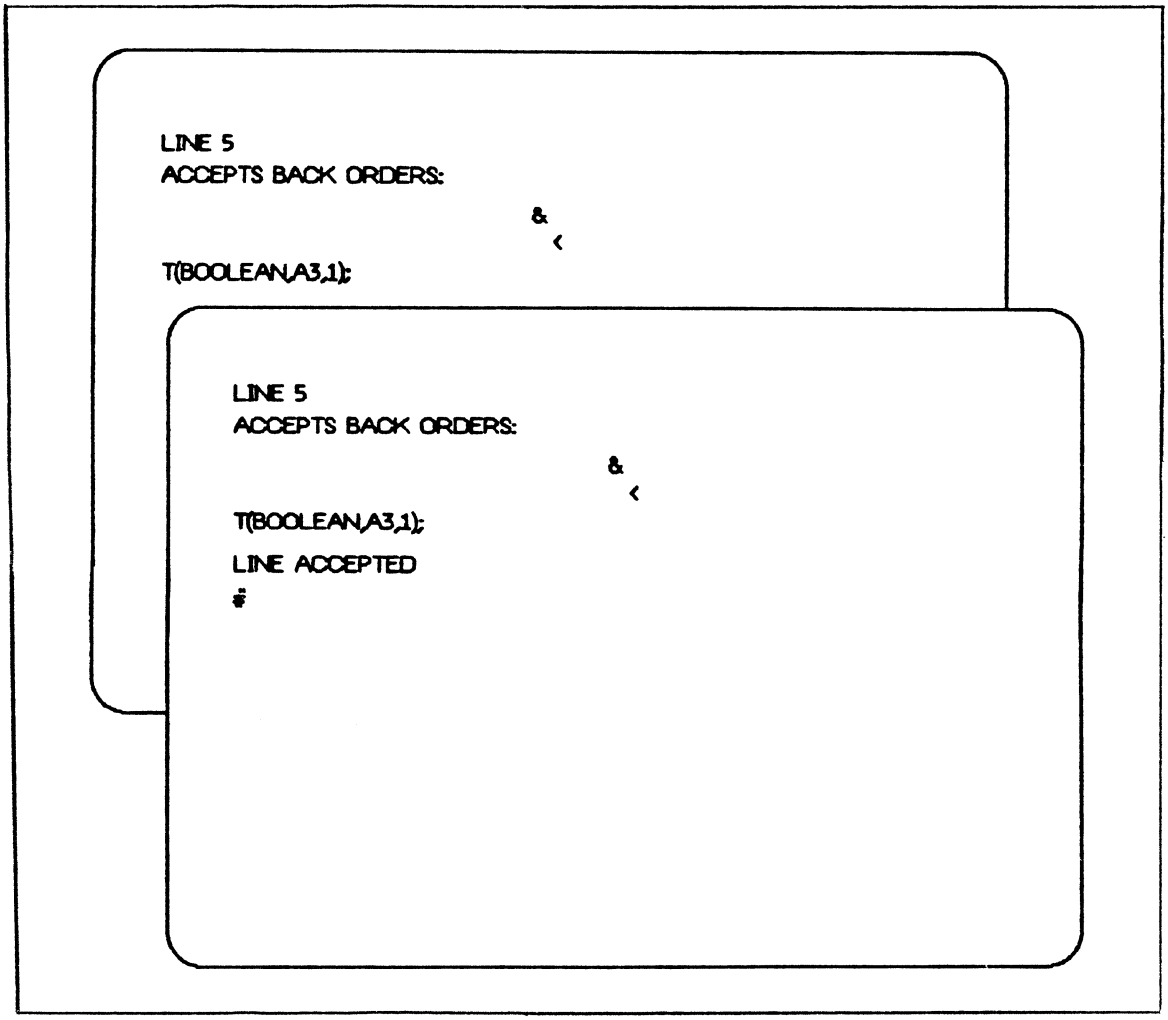


Figure 4-7. Line Insertion - TCL Function

COMPILING A FORMAT

A format can be compiled at any time after a format layout is received. The compilation of a format is a two-step process. The first step is to translate the Format Layout file to TCL source images. These images are put in the TCL work file. The second process is to compile these TCL work file images. The resulting format is stored in the Test Format file. Compiling can be manual or automatic. This is determined by the AUTOCOMPILE attribute of the FORMAT command.

If AUTOCOMPILE is set (or allowed to default) to FALSE, then compilation is manual. In this case, the user must enter the commands for translation (TRANSLATE) and compilation (COMPILE <format name>) manually, after the format layout is accepted.

If AUTOCOMPILE is set to TRUE, then compilation follows automatically. After the format layout is transmitted and accepted by the Format Generator, the format is automatically translated and compiled. The user is informed that the format has been translated and compiled.

Whenever translation is performed, the TCL source images are added to the end of the TCL work file. If the format already exists in the work file, it is removed. If the compile contains errors, the errors are displayed on the terminal, and the compile is stopped before completion.

Figure 4-8 illustrates automatic compilation of the format Customer starting with the format layout. Assume AUTOCOMPILE was set to TRUE in the FORMAT command. The first screen shows the user's entry; the second indicates that the format layout was accepted. The third indicates that translation is complete, and the fourth tells the user that the compilation was completed successfully.

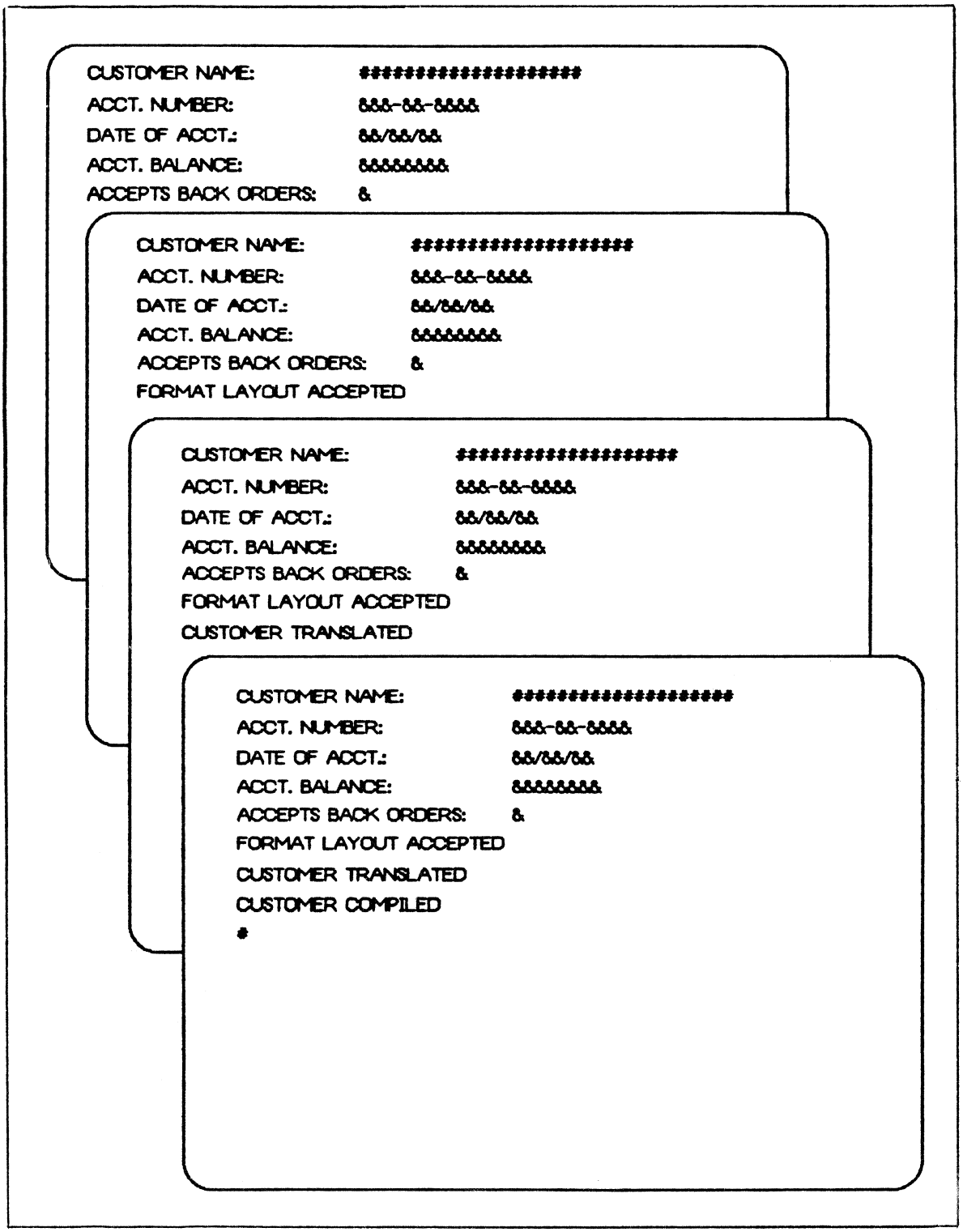


Figure 4-8. Automatic Compilation of a Format

TESTING A FORMAT

Once a format is compiled into the Test Format file, it can be tested. A format can be tested as an input or an output format. A format directed to a terminal is an output format. Data can be supplied for the format, or it can be treated as a forms request (for example, no data).

Figure 4-9 illustrates testing the Customer output format. The first screen shows the user's entry of the TESTOUT command with data. The second screen displays the format with data applied as it appears in a "live" environment.

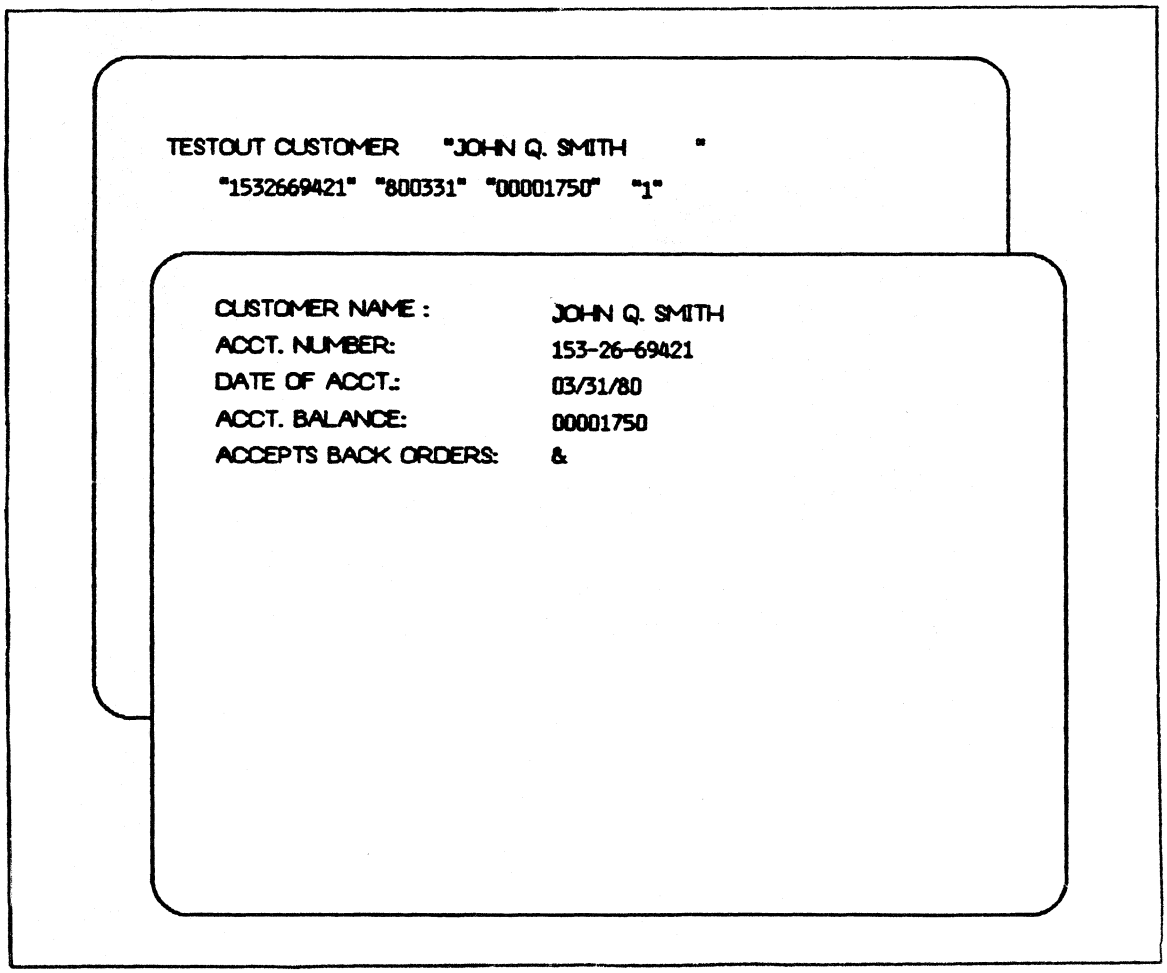


Figure 4-9. Testing a Format

RETRIEVING A FORMAT

An existing format can be retrieved for subsequent processing by using the GET command. A format can be retrieved from either the user's Test Format file or from any GEMCOS format file. To do this, a format must already exist in the specified file. The default file is the Test Format file. Upon successful completion of a GET command, the Format Layout file contains a picture of the retrieved format with any necessary line insertions already made.

The GET command is used mainly for testing and modifying existing formats. However, by translating the format layout after executing a GET command, a copy of the TCL source images for the retrieved format can be obtained.

Figure 4-10 illustrates retrieving the Customer format from the Live Format file MCSFORMATS. This format can now be compiled into the Test Format file and tested. As shown, this format was originally compiled with a clear-screen control character.

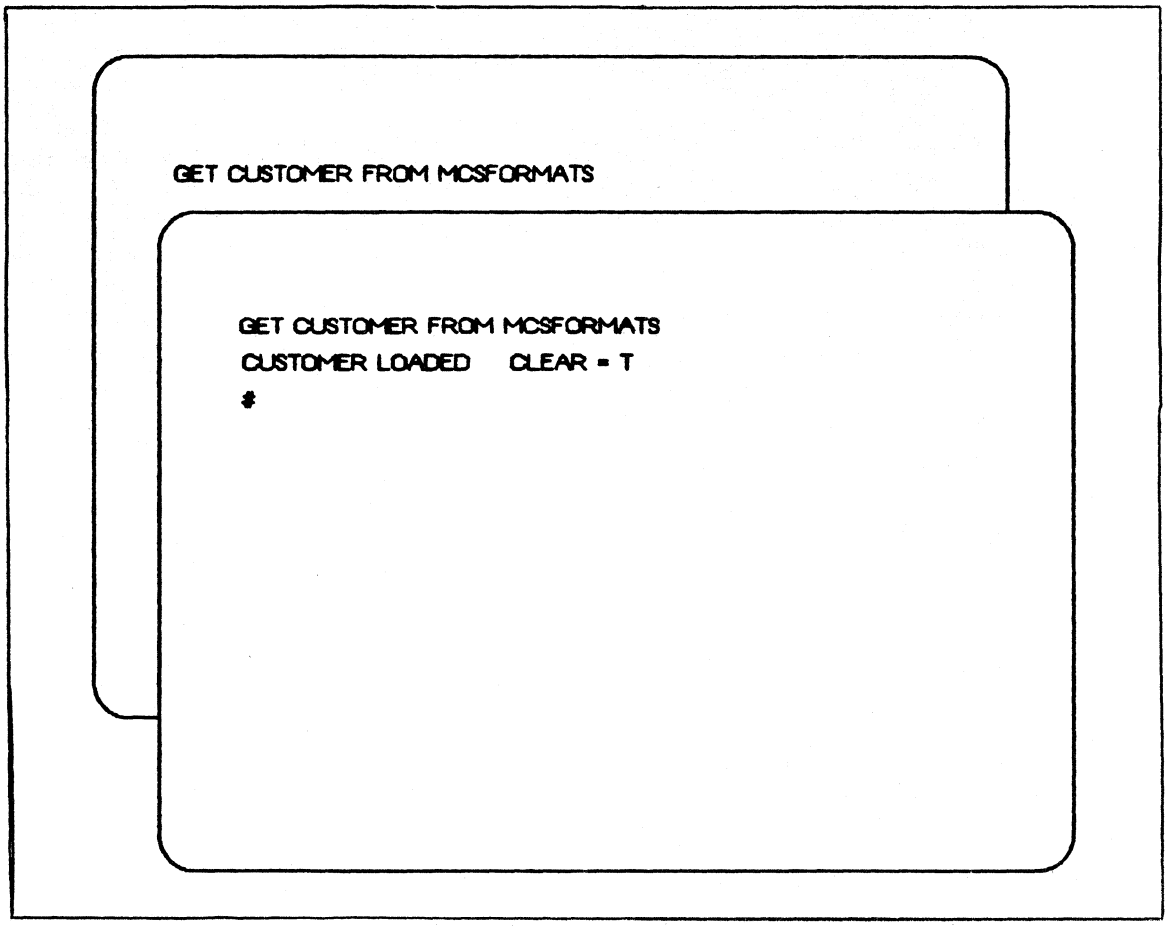


Figure 4-10. Retrieving a Format

If the screen size of the user's terminal is not 24 lines by 80 characters, it is necessary to inform the Format Generator of the terminal's dimensions. This is so the Format Generator can allocate carriage returns correctly and compute line boundaries accurately. The TERMINAL command is used to accomplish these changes.

Figure 4-11 illustrates the use of the TERMINAL command to inform the Format Generator that the user's terminal is a TD700. Since the screen size of that terminal is 8 lines by 32 characters, the user's entry on screen 1 accomplishes the necessary reassignment. Screen 2 shows the Format Generator's response to this command. The last line of the response indicates that the user is currently using the system Defines file.

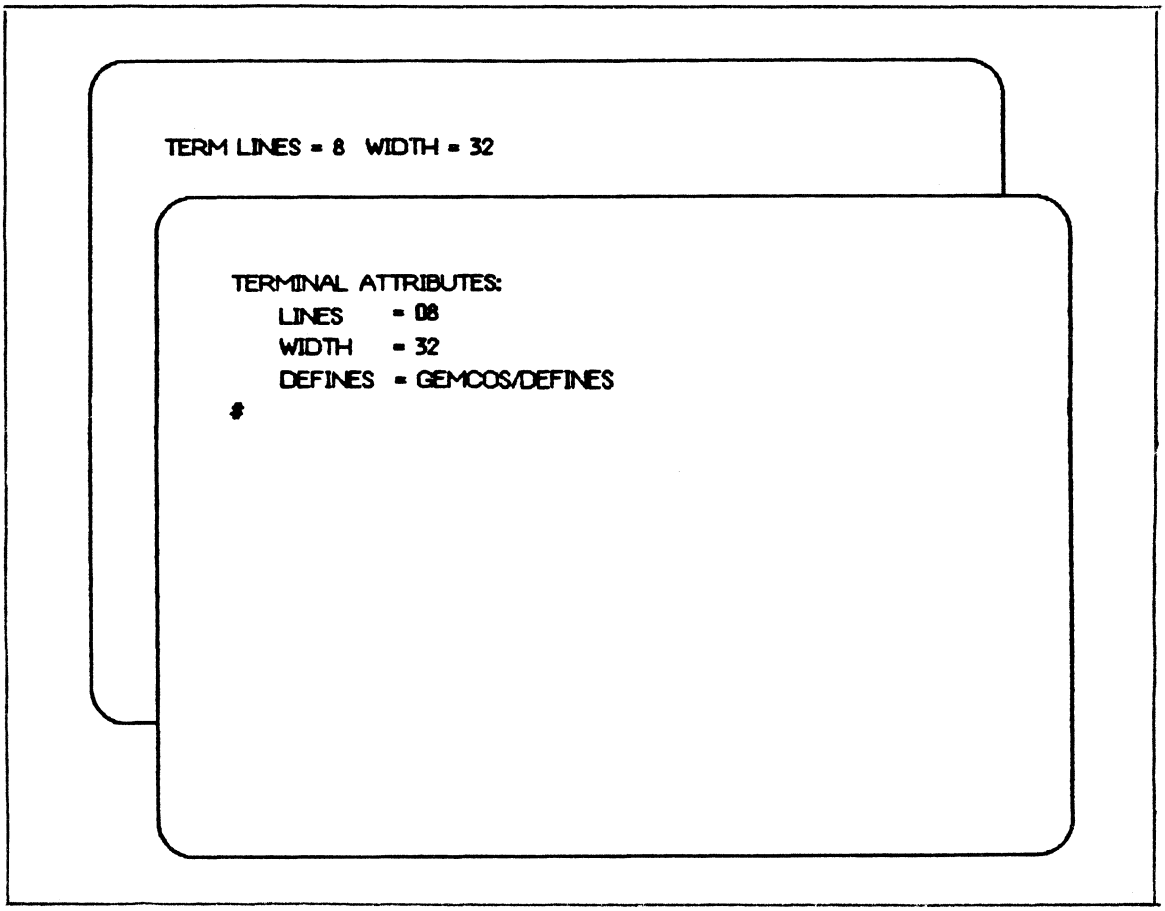


Figure 4-11. Changing Terminal Attributes

DISPLAYING A FORMAT

Two commands can be used to display the current contents of the Format Layout file. The first command, DISPLAY, sends the format layout to the terminal. The second command, WRITE FORMAT, sends the format layout to the line printer. In both cases, only the literals and data are displayed. Line-insertion information is not displayed with these commands. These commands are useful when a temporary or permanent display of the format is required.

Figure 4-12 demonstrates the DISPLAY command. In this example, it is assumed that the Format Layout file contains the format Customer, as in previous examples.

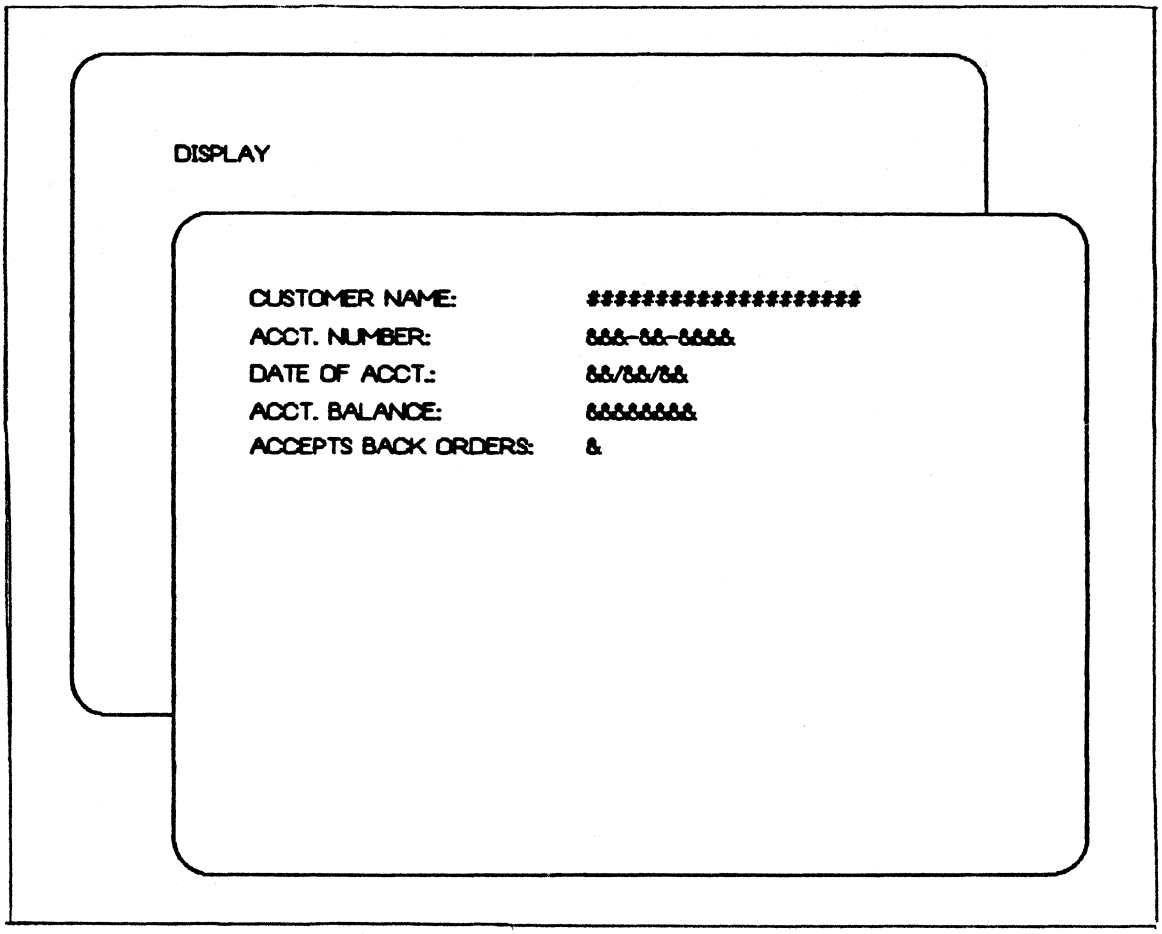


Figure 4-12. Displaying a Format

UPDATING A FORMAT

It is often desirable to modify a format from a live GEMCOS network while the MCS is running. Once a format is compiled into the user's Test Format file, the update mechanism allows the user to modify any format in any GEMCOS format file whether or not the MCS is running. The UPDATE command or the update feature of the COMPILE command takes a compiled format from the user's Test Format file and places it in any Live Format file, provided the format exists in the specified live file.

An example of the update process follows:

It is necessary to modify one of the lines of the Customer format in the GEMCOS format file MCSFORMATS. The steps involved are:

1. Get the format from the GEMCOS format file and convert it into a format layout.

2. Modify the line in question.

3. Translate and compile the format.

4. Transfer the format back to the live format file.

These steps use the following commands:

1. GET CUSTOMER FROM MCSFORMATS
2. <LINE command to modify line in question.>
3. TRANSLATE
4. COMPILE CUSTOMER
5. UPDATE TO MCSFORMATS CUSTOMER

This updated format becomes available to the live GEMCOS network when the GEMCOS network control command UPDATE (UPD) is executed through the GEMCOS MCS. Note that for this example, the COMPILE and UPDATE commands could be replaced with the following command:

COMPILE UPDATE TO MCSFORMATS CUSTOMER

MODIFYING THE TCL WORK FILE

The following procedures are used to manipulate the the TCL work file. They allow the user to add and delete records from the work file, display records, and save on disk the content of the work file.

MODIFYING A TCL WORK FILE RECORD

Modifying a record in the TCL work file involves two commands, LIST and SEQUENCE. Figure 4-13 illustrates the two-step process of modifying a TCL work file record.

Screen 1 shows the user's request to display the TCL work file record that has sequence number 00006000 (that is, the LIST command). Screen 2 shows the requested record returned by Format Generator.

Note that sequence numbers are eight characters long and occupy character positions 73 through 80 of the work file record. The sequence number is displayed at the beginning of the line.

The user can now make the necessary changes to the line and send it back to the Format Generator using the SEQUENCE command, shown in screen 3. Finally, screen 4 indicates the Format Generator's response. The line is modified and available for processing.

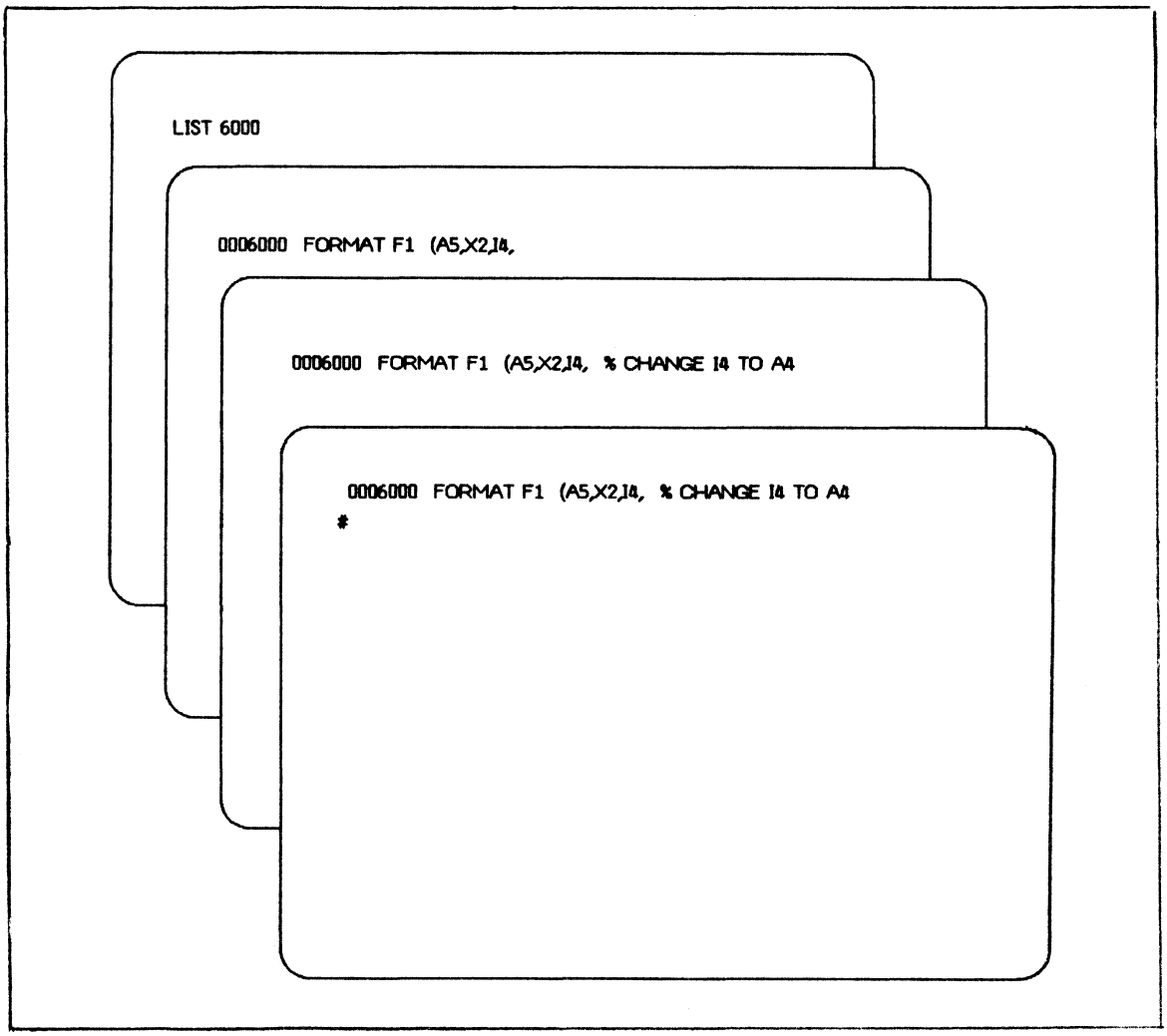


Figure 4-13. Modifying a TCL Work File Record

MERGING TCL WORK FILE RECORDS

There are two commands available for merging records to the TCL work file: MERGE and RMERGE. They differ only in the handling of duplicate records (refer to explanations of the MERGE and RMERGE commands). The basic function of these commands is to add existing TCL format and function records into the TCL work file. Figure 4-14 illustrates the merging of records into the work file. USER1/TCL.SPECS is a disk file containing the entire TCL specifications for USER1's MCS.

In this example, only the function and format statements are merged in the work file. Merged records retain their sequence numbers. These added functions and formats can be compiled at the user's convenience.

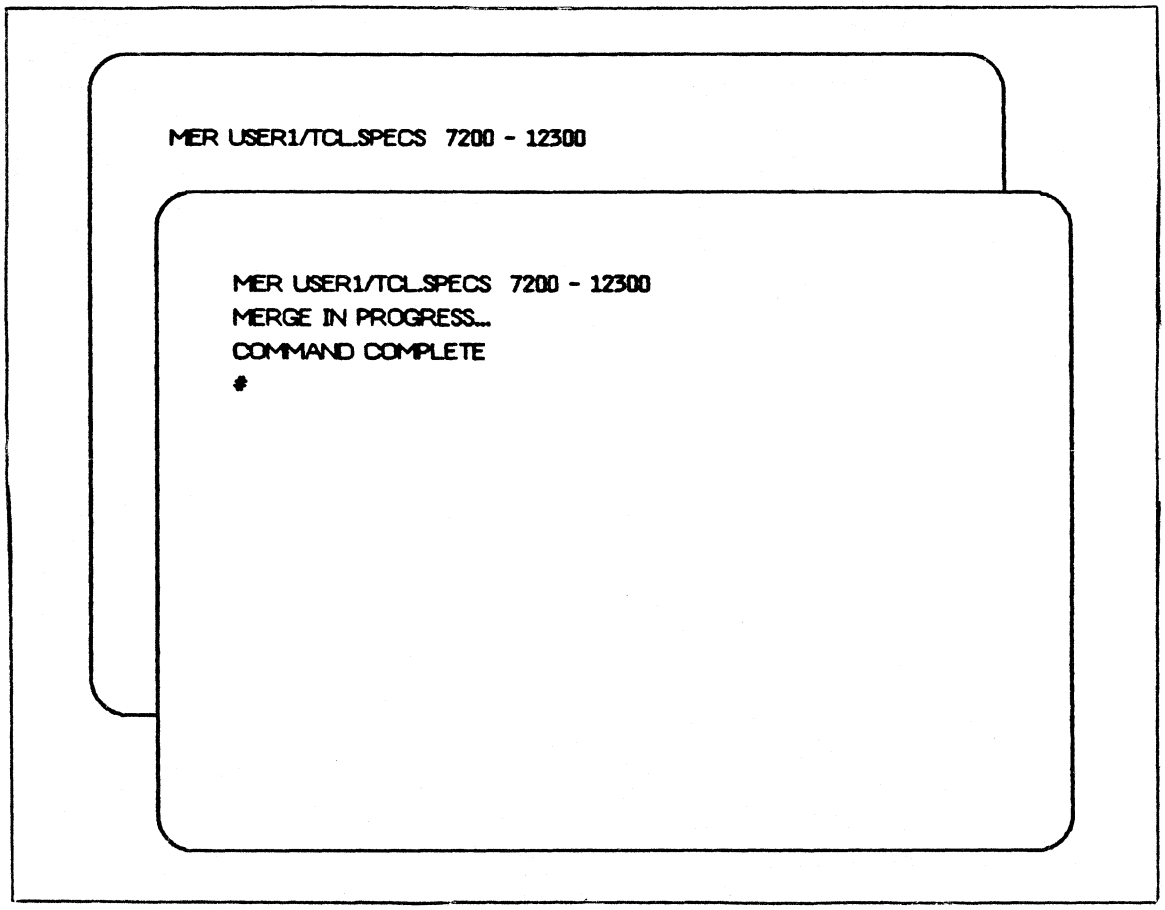


Figure 4-14. Merging TCL Work File Records

SAVING THE TCL WORK FILE

Though normally the TCL work file is purged at the end of the session, the contents of the TCL work file can be saved on disk. The saved file contains all the current functions and formats created or altered in the work file. The contents of the saved file can be merged into the main TCL specifications file. Figure 4-15 illustrates the use of the SAVE command. Upon receipt of the Format Generator's response, the disk file USER1/TCL.NEW is locked in the disk directory.

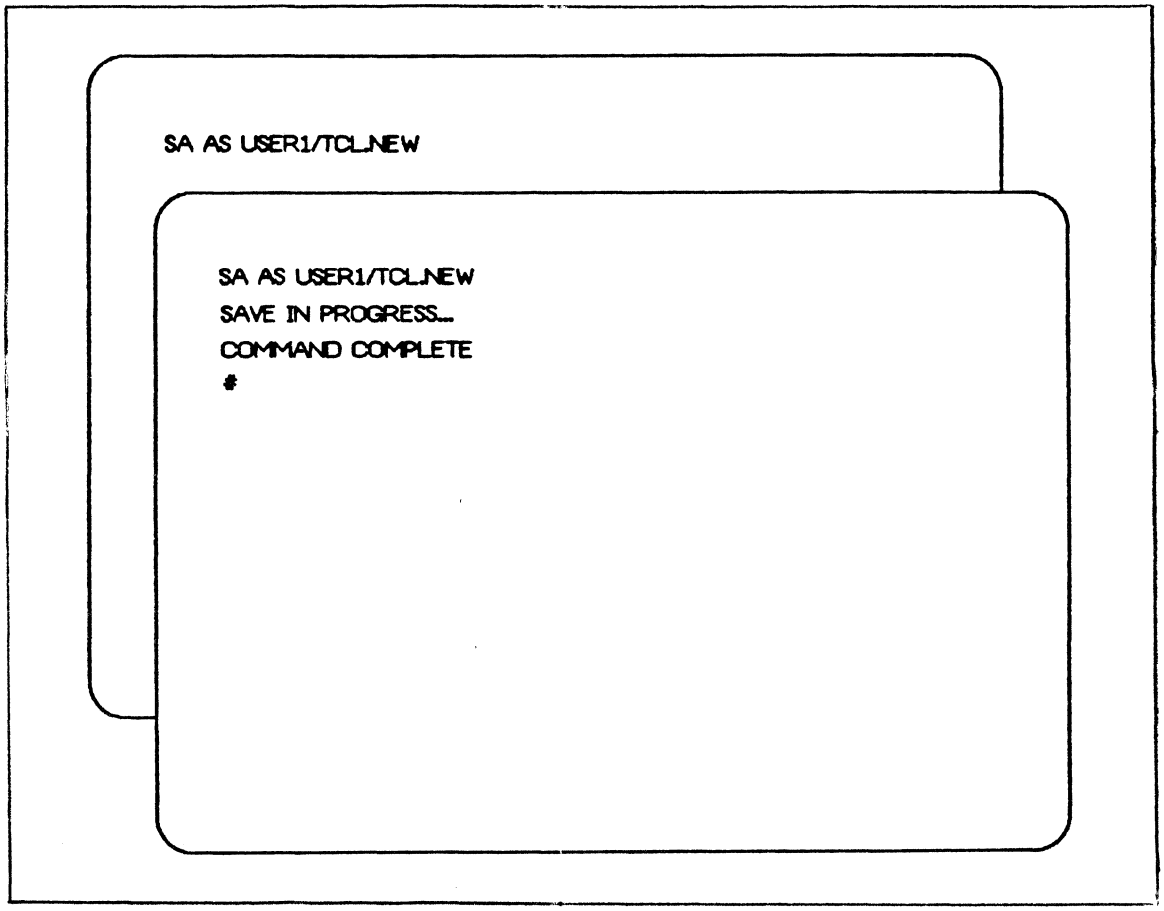


Figure 4-15. Saving the TCL Work File

SECTION 5

FORMAT GENERATOR COMMANDS

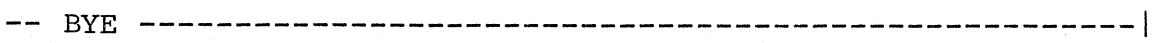
This section provides a detailed description of the complete set of Format Generator commands. "Railroad" syntax diagrams are used to describe the syntax of these commands. An explanation of syntax conventions can be found in the GEMCOS Reference Manual. Commands are treated in alphabetical order.

The following is a list of all the Format Generator commands.

BYE
COMPILE
DEBUG
DEFINE
DELETE
DISPLAY
FORMAT
FORMAT LAYOUT
GET
HELP
IGNORE
LINE
LIST
MERGE
NULL INPUT
RESEQ
RMERGE
SAVE
SEQUENCE
STATUS
TEACH
TERMINAL
TEST
TRANSLATE
UPDATE
WHAT
WRITE

BYE

Syntax:



Semantics:

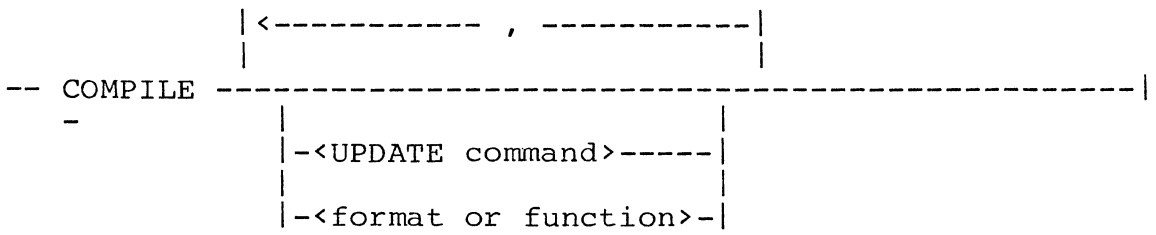
The BYE command terminates the user's current formatting session and purges the user's TCL work file, the Format Layout file, and the Test Format file. When the command is entered by the last active user, the Format Generator goes to end-of-job (EOJ).

Example:

```
BYE
```

COMPILE

Syntax:



Semantics:

The COMPILE command causes the Format Generator to compile all or part of the user's TCL work file. This command allows syntax checking of functions and formats. Those functions and formats that compile successfully are placed in the user's object Test Format file. They may be tested using the TEST command. If AUTOCOMPILE is set to true in the FORMAT command, the COMPILE command is automatically executed upon receiving a format layout.

When the UPDATE command is used, an UPDATE is automatically performed on those formats that compile successfully. Refer to the discussion of the UPDATE command for a detailed description of the UPDATE feature. If a compile is automatically performed because AUTOCOMPILE is set to TRUE and the Format Generator is not executed in UPDATE mode (UPDATE is the default), no update is performed.

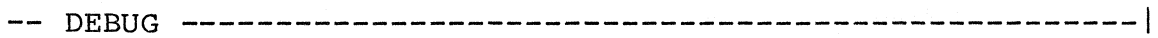
If no format or function name is specified, all functions and formats in the Transaction Control Language (TCL) work file are compiled. Otherwise, only those functions and formats specified are compiled. If a specified function or format is not in the TCL work file, an error results, and that function or format is not compiled.

Examples:

```
COMPILE
C FMT1, FMT2
C UPDATE
C UP TO TEMP/FMTFILE FMT2, FMT4
```

DEBUG

Syntax:



Semantics:

The DEBUG command is used as a toggle which alternately sets and resets debugging logic built into the Format Generator. It is not used for creating or testing formats but for evaluating the performance of the Format Generator code itself.

When the debugging logic is invoked, the following information is written to a printer file:

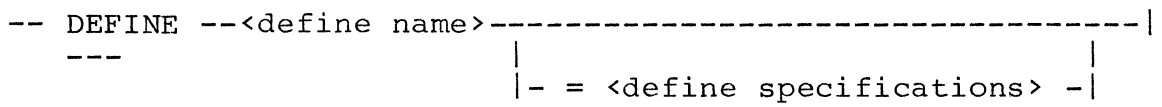
- A monitor of all procedures entered, including pertinent data for that procedure.
- A program dump. This occurs whenever the Format Generator encounters a serious or fatal error.

Example:

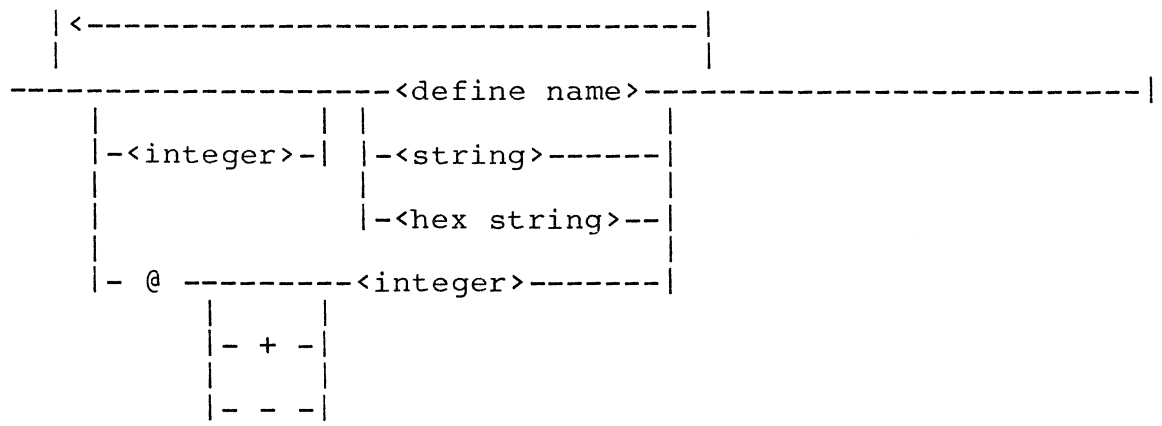
```
DEBUG
```

DEFINE

Syntax:



<define specifications>



Semantics:

The DEFINE command can be used to represent the characters required to put a terminal into FORMS mode. Rather than refer to control strings by their hexadecimal values, the user may find it more convenient to introduce a special shorthand for frequently used strings.

For example, the control string 4"27E6" enables certain terminals to go into FORMS mode. However, this control string may vary from terminal to terminal and is difficult to remember. The define DEFINE FORMSMODE = 4"27E6" can be used to represent the characters required to put a terminal into FORMS mode. The user enters only FORMSMODE to invoke this terminal function.

TCL location specifications can also be used in a define, either alone or mixed with control strings. Define specification integer values must be less than 65536. A define name can be a maximum of 17 characters in length.

Using an integer within the define specifications clause allows for multiple occurrences of a define name or any string. The allowable values are between one and 255, with one being the default value. Defines can be nested to a maximum of nine levels deep.

When a user enters the DEFINE command, the define named is added to the current Defines file for the terminal. The default Defines file is named GEMCOS/DEFINES and is shared by all users. Initially, this list contains predeclared defines. As new defines are added to the file, either replacing or supplementing existing ones, they become available to all users. If a user wants to use a Defines file other than the default Defines file, the current file can be changed using the define specifications option of the TERMINAL command (refer to the discussion of the TERMINAL command).

The user specifies the place on a format layout where he/she wants a define used. This is done with the LINE command. Refer to the discussion of the LINE command for a more detailed description of define usage.

The default Defines file contains certain predeclared defines. The following list gives the names, hexadecimal values, and meanings of the predeclared defines:

Define Name	Hex Value	Meaning
CAN	4"18"	Start blinking (hex value requires one character position).
CR	4"0D"	Carriage return.
DC1	4"11"	Stay in receive.
DC4	4"3C"	Cursor home.
EM	4"19"	Protected data (hex value requires one character position).

ESC	4"27"	Escape.
FF	4"0C"	Form feed.
HT	4"05"	Horizontal tab (hex value requires variable number of character positions).
LF	4"25"	Line feed.
NULL	4"00"	Null.
SI	4"0F"	Start underline (hex value one character position).
SO	4"0E"	Start negative video (hex value requires one character position).
SUB	4"3F"	Start bright video (hex value requires one character position).
CLEAR	FF	Clear screen (hex value requires one character position), home cursor.
FORMSMODE	ESC "W"	Enable FORMS mode.

When DEFINE <define name> appears alone, the define name and its current value are displayed. If the specified define does not exist, an error message results. Hex value requires a variable number of character positions.

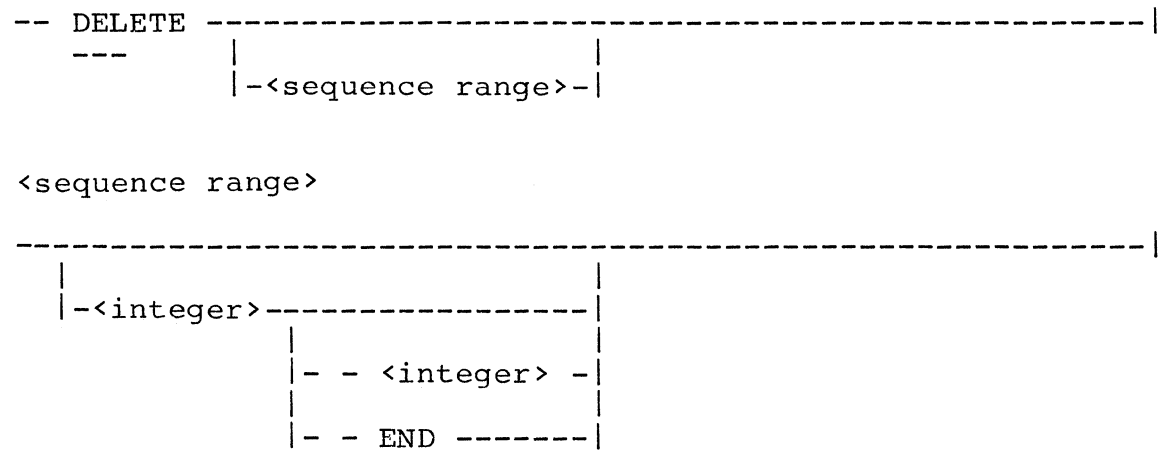
Examples:

```
DEFINE CLEAR = FF
DEFINE NEGBLINK = SO CAN
DEFINE LINE80 = 80 "-"
DEFINE SMALLA = 4"81"
```

```
DEFINE SKIP4 = @ + 4
DEFINE FORMSMODE
```

DELETE

Syntax:



<sequence range>

Semantics:

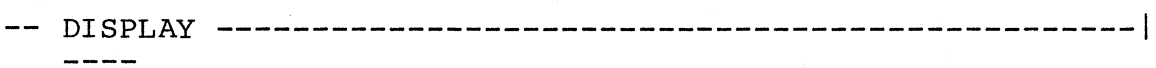
The DELETE command allows deletion of single lines, ranges, or of the entire TCL work file. It affects only the TCL work file.

Examples:

```
DEL 100 - 500
DEL 2000 - END
DEL 300
DELETE   %(deletes entire TCL work file)
```

DISPLAY

Syntax:



Semantics:

The DISPLAY command is used to display the contents of the Format Layout file. Only the text and data portions of each line are displayed. Location and control information can be viewed or entered using the LINE command.

The DISPLAY command is often used in the following situation. A format is received as a result of entering the FORMAT command followed by a format layout. However, the user wants to change the layout. The user re-enters the FORMAT command, placing the terminal in an Expecting Format Layout mode. Instead of laying out an original format, the operator enters DISPLAY to obtain the current layout for correction. The terminal is still in Expecting Format Layout mode, and the operator can now correct the layout and return it to the Format Generator.

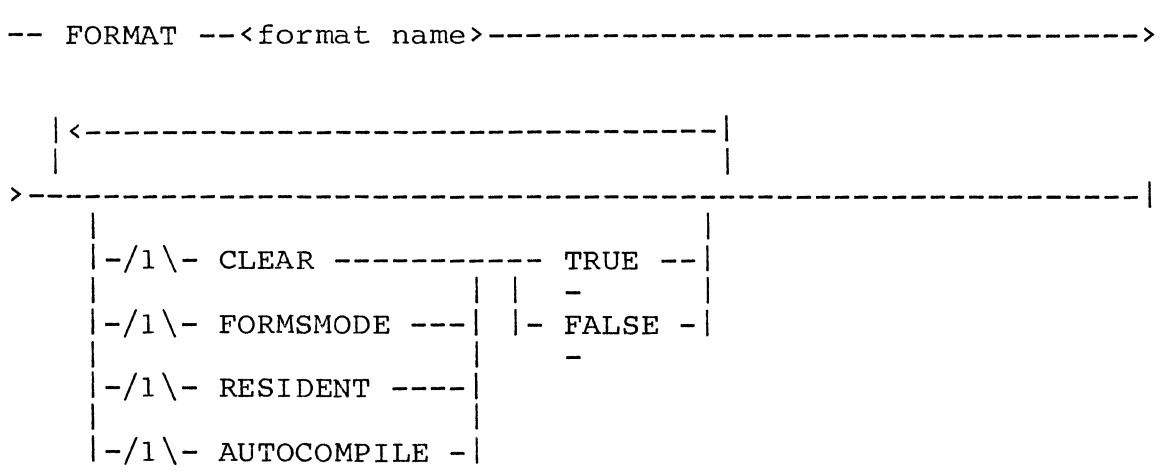
If the DISPLAY command is used in this manner, all location and control information previously associated with the format is discarded when the format is re-sent. If this command is entered while the terminal is in an Expecting Format Layout mode, the word DISPLAY must be entered starting at the first position on the terminal. It must be the only word entered.

Example:

```
DISPLAY
```

FORMAT

Syntax:



Semantics:

The FORMAT command is used to identify and establish certain attributes for a format to be created. (See "Format Layout" in this section.) Unless there is an attribute error found in the input, the FORMAT command always leaves the terminal in an Expecting Format Layout mode. This means that a format has been named and the Format Generator is now expecting a layout of the format from the next transmission. Once in Expecting Format Layout mode, the user must then do one of three things:

- Clear the screen, lay out a format on the screen and transmit the format layout to the Format Generator.
- Enter IGNORE, which removes the terminal from Expecting Format Layout mode. Then enter another command, or start over with a new format name and attributes.
- Enter DISPLAY, which displays a previously created format layout for correction and leaves the terminal in an Expecting Format Layout mode.

To determine if the terminal is in an Expecting Format Layout mode, a STATUS or WHAT command can be entered without changing the current format or terminal status.

Format attributes are assigned logical values of TRUE or FALSE. A format name can be a maximum of 17 characters in length.

The format attributes have the following meanings:

ATTRIBUTE	DESCRIPTION
CLEAR	Inserts the contents of the CLEAR define at the beginning of the format. The default value of the CLEAR define is 4"0C". The default value of the CLEAR attribute is FALSE.
FORMSMODE	Inserts the contents of the FORMSMODE define at the end of the format. The default value of the FORMSMODE define is 4"27E6". The default value of the FORMSMODE attribute is FALSE. Any allowable characters are acceptable as forms characters. The symbols US, GS, and FS may be used to start a data field, and the RS symbol may be used to end a data field.
RESIDENT	Causes the format to be compiled as a format format. Since resident formats are kept in memory rather than disk, their response time is generally faster than that of nonresident formats. To make a resident format available to the "live" GEMCOS environment, the MCS has to be restarted after the format is updated. (Refer to the discussion of the UPDATE command.) The default value for the RESIDENT attribute is FALSE.
AUTOCOMPILE	Causes the TRANSLATE and COMPILE commands to be executed automatically by the Format Generator as soon as the format layout is received. If the translated format is a new format, it is added to the end of the TCL work file; if not, the translated format

replaces the old copy. The compiled format is added to to the user's object Test Format file. The default value for the AUTOCOMPILE attribute is FALSE.

Examples:

```
FORMAT FMT1
FORMAT FMT2 CLEAR=T FORMSMODE=T RESIDENT=T AUTOCOMPILE=T
```

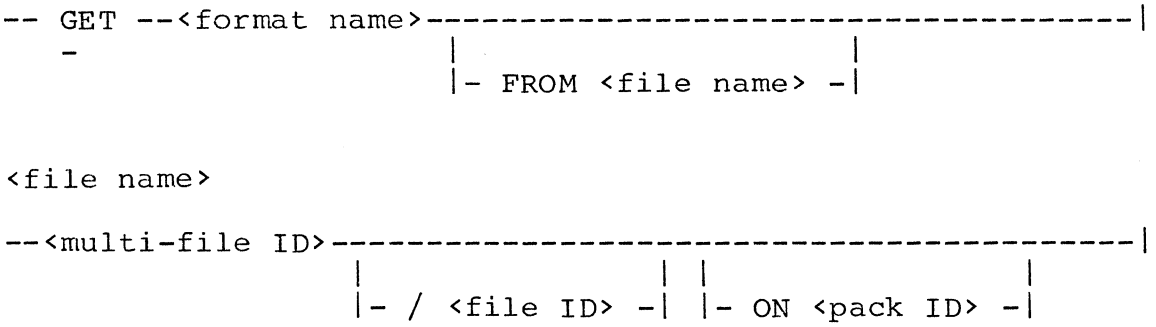
FORMAT LAYOUT

The format layout is not a command, but simply the screen layout of a format transmitted to the TCL Workfile when the terminal is in Expecting Format Layout mode.

Refer to "Creating a Format" in Section 4 for instructions on how to lay out a format.

GET

Syntax:



Semantics:

The GET command is used to retrieve a format from an object format file and to convert the format into a format layout. The converted format is stored in the Format Layout work file. If there is no file name specified, the Format Generator attempts to retrieve the format from the user's current object Test Format file.

The following attributes cannot be retrieved from an object format:

- Update variables in a variable repeat expression.
- If there is a file name specified, TCL functions are not converted.

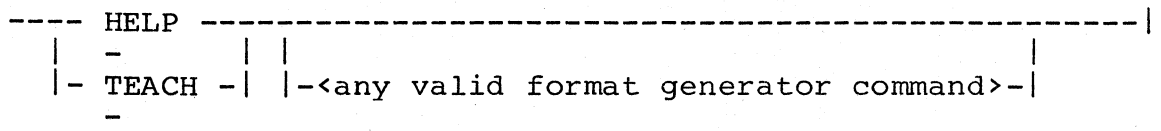
Successful conversion of the specified format results in a format layout as though the user had used the FORMAT and LINE commands to layout and modify the format from scratch.

Examples:

```
GET FORMATX1
G FORMATX2 FROM FORMAT/LIBRARY
```

HELP

Syntax:



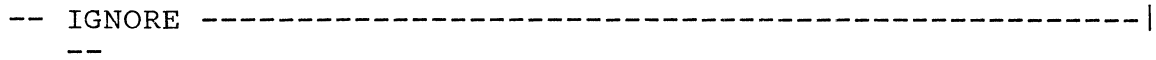
The HELP command is used to obtain a list of valid Format Generator commands or to learn the syntax of a particular command. Refer to the discussion of the TEACH command for more detail.

Example:

```
HELP
```

IGNORE

Syntax:



Semantics:

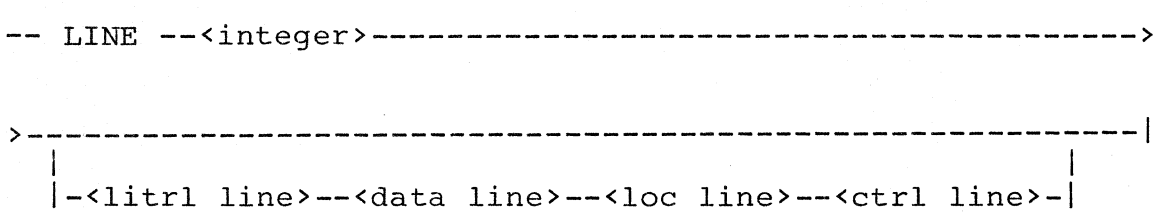
The IGNORE command terminates the Expecting Format Layout mode. This command must be entered starting at the first position on the terminal, and IGNORE must be the only word entered.

Example:

```
IGNORE
```

LINE

Syntax:



Semantics:

The LINE command performs two primary functions:

- It allows for the insertion of the pound sign (#) and the ampersand (&) as text characters within literal expressions on a layout. These characters cannot be used as text characters in a format layout due to their special usage in defining fields.

- It allows for the insertion of terminal control strings into the format layout. This type of insertion is useful when the format layout requires characters not available on the user's keyboard.

When the LINE command consists only of the word LINE followed by a line number, the current layout of the line is displayed for modification.

An explanation of the four different types of lines and their associated meanings follows:

Line Type	Explanation
<literal line>	Contains the titles and headings which are a part of the message format. These literals become part of the message. An An example of the first usage of the LINE command is demonstrated in the

following example. The user wants the format to read ID #:, but has to enter ID #: because the pound sign (#) is a special control character. However, the pound sign (#) can be inserted in the proper place on the literal line.

<data line> Each character on the data line represents a data character in the program message. The special characters (pound sign (#) and ampersand (&)) originally were inserted using the FORMAT command. These special characters can be added or deleted on this line wherever necessary.

<location line> Used to specify the position within the line where a control string is to be inserted. A left angle bracket (<) indicates that a control string clause is to be inserted preceding this character position. A right angle bracket (>) indicates that a control string clause is to be inserted following this character position. For each left or right angle bracket in the location line, there must be a corresponding control string clause on the control line.

<control line> Consists of define names, control strings, and location specifiers (control specs) starting with an at sign (@). Refer to "Location Specifiers (Remapping)" in Section 3 for an explanation of location specifiers. A control string clause can consist of any number of control specs. However, control specs clauses must be separated by commas.

An example of a displayed response to entering LINE 1 is shown Figure 5-1.

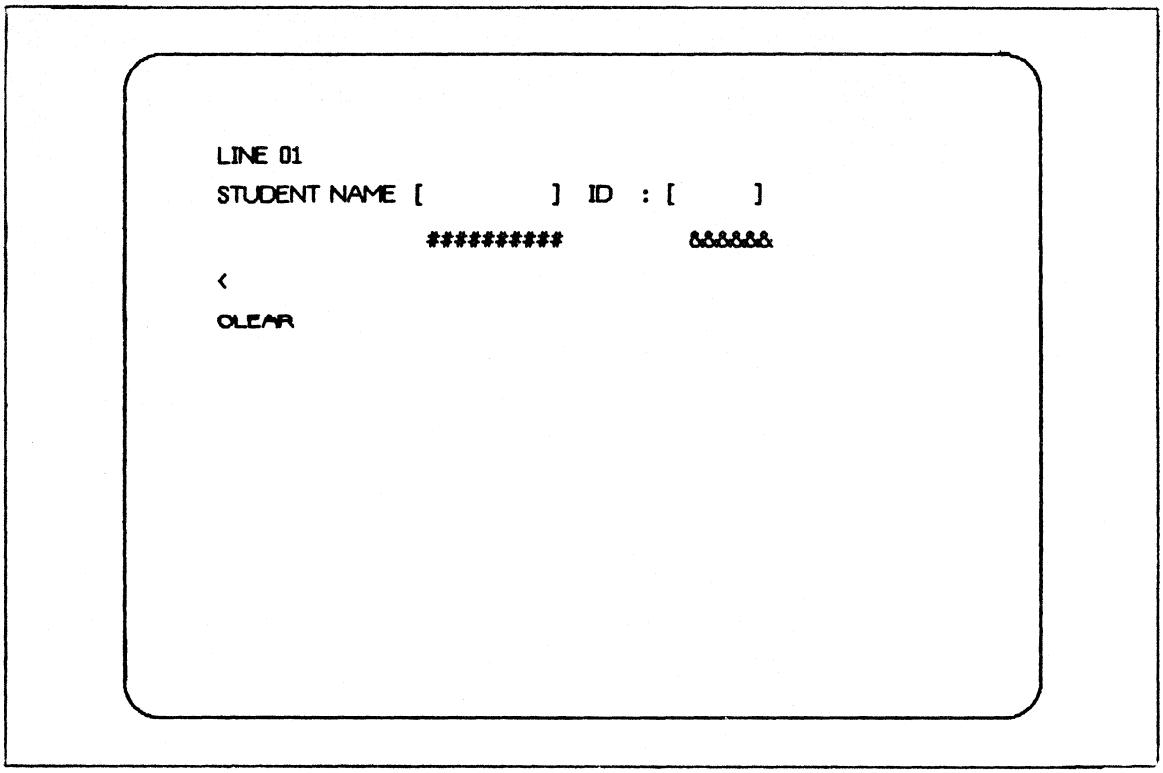


Figure 5-1. Response to Entering LINE 1

Example:

In Figure 5-1, the user wants to make the following changes:

1. Add a pound sign (#) after the word ID. This is done by inserting the pound sign (#) at the appropriate place on the <literal line>.

2. Modify the CLEAR function to clear, followed by a horizontal tab. This is done by changing CLEAR to CLEAR, HT on the <control line>.

3. Adjust the internal location pointer in the data message prior to extracting the student name from the message. This is done by inserting a right angle bracket (>) on the location line underneath the first left square bracket ([), and inserting

the sequence "; @ +8" at the end of the <control line>.

After this modification is completed, Figure 5-1 appears as Figure 5-2 prior to transmission:

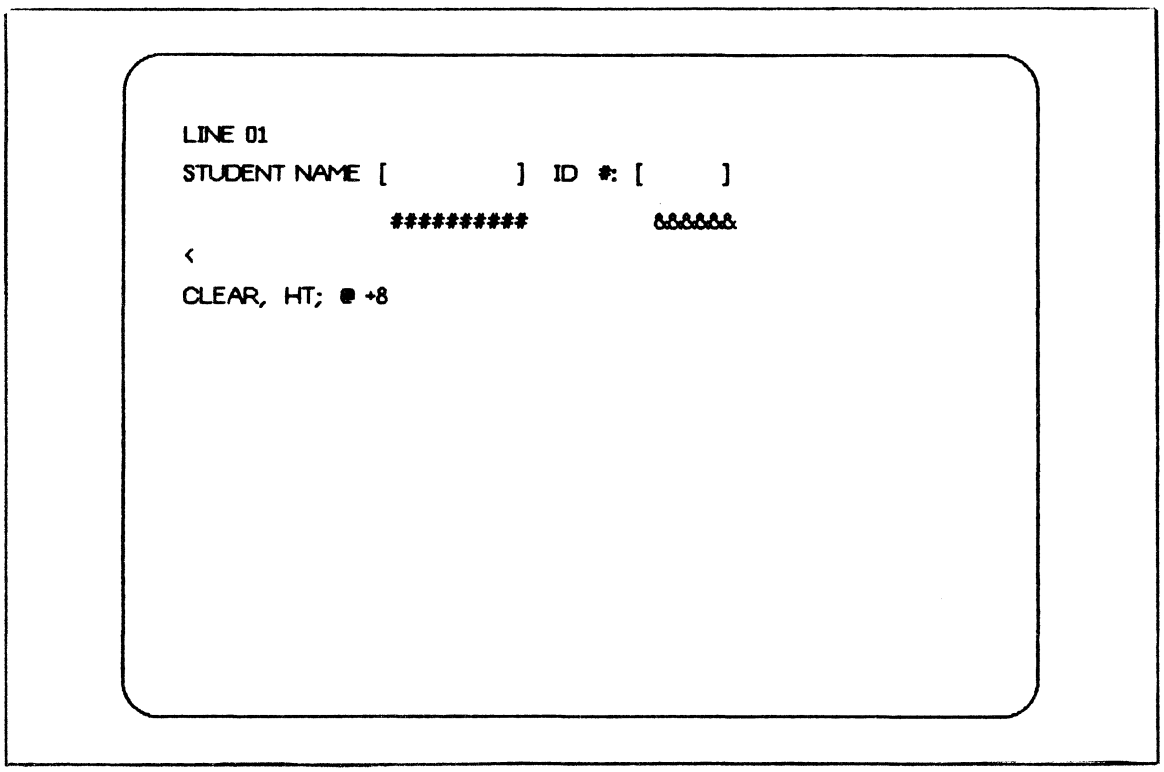
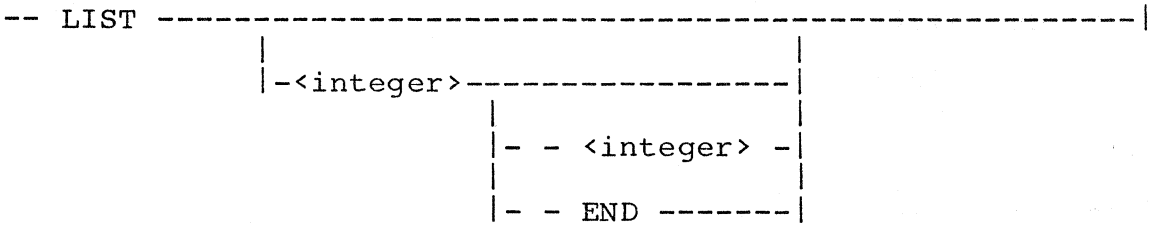


Figure 5-2. Modified Response to Entering LINE 1

LIST

Syntax:



Semantics:

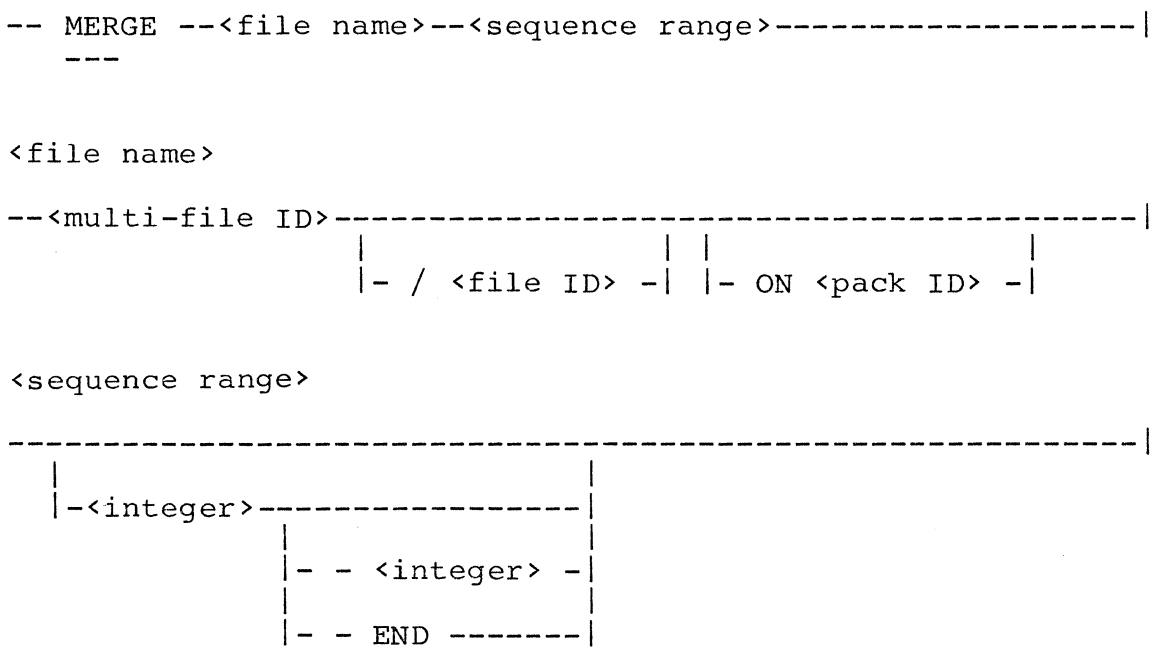
Diverse information can be displayed using the LIST command. If LIST is entered alone or with a sequence range, all or portions of the TCL work file are displayed. If L(IST) DEFINES is entered, a list of the names of all defines in the user's current Defines file is displayed. If L(IST) FORMATS is entered, a list of the names of all formats in the user's Test Format file is displayed. If the list involves more than one full screen of data, subsequent screens can be displayed by transmitting a blank or series of blanks. The list can also be terminated by the entry of any other command.

Examples:

```
LIST
L 100 - 600
LIST DEFINES
L FORMATS
```

MERGE

Syntax:



Semantics:

The MERGE command merges all or part of the specified file with the TCL work file by collating the sequence numbers. The merged records become part of the TCL work file. When records in the merged file and the TCL work file have identical sequence numbers, the TCL work file records are retained, and the merged file records are ignored.

Examples:

```
MERGE PERMANENT/TCL
MER TEMP/TCL.BR 1200-2500
```

NULL INPUT

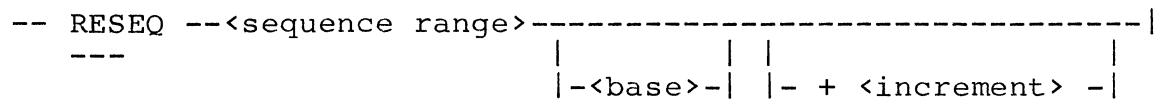
Null input is not a command. It consists of a series of one or more blanks transmitted from home position. A null input is used only to receive the next screen of requested information after a LIST command is entered.

Example:

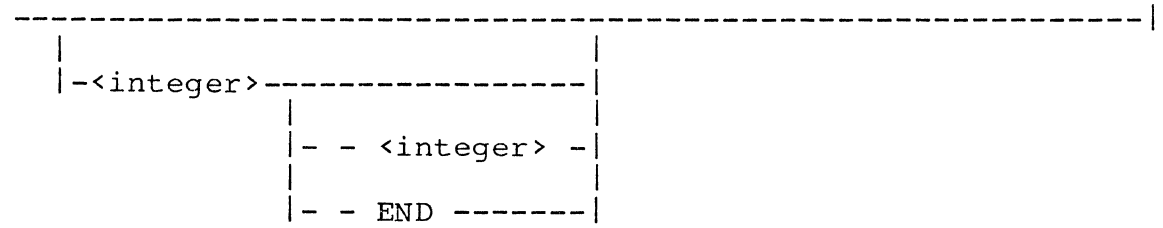
<one space>

RESEQ

Syntax:



<sequence range>



Semantics:

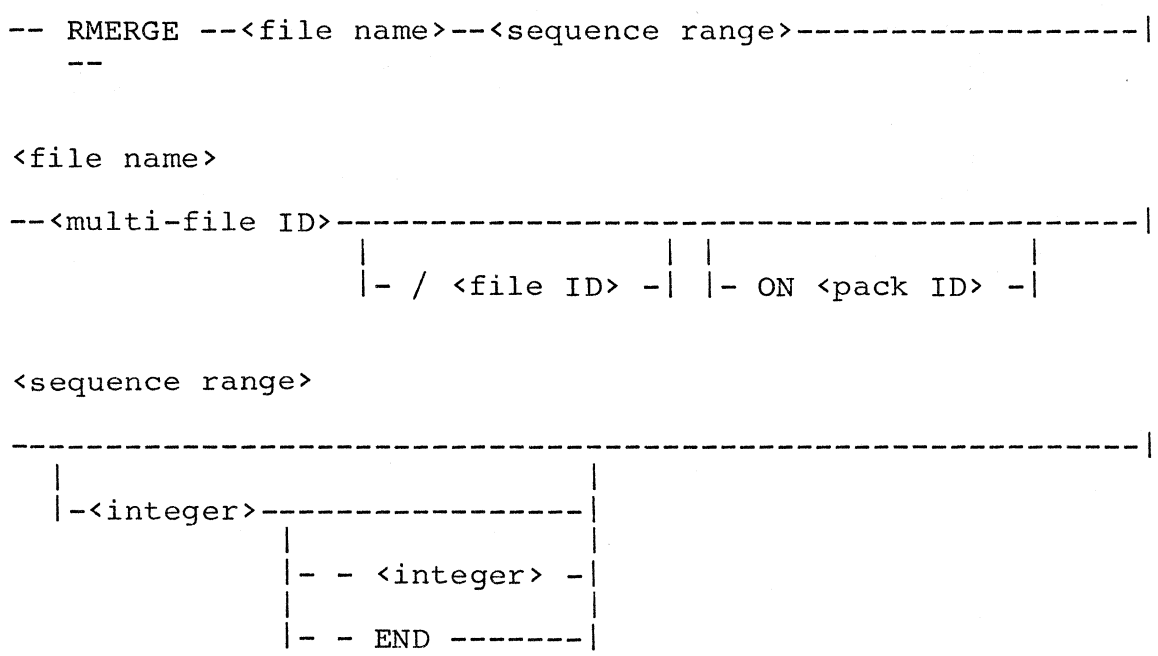
The RESEQ command is used to resequence all or part of the TCL work file. This command affects only the TCL work file. If the no sequence range is specified, the entire TCL work file is resequenced. If no base is specified, the default value of 100 is used. If no increment is specified, the default value of 100 is used. If a sequence range is specified, then the resequenced range must not exceed the original range specified. This command should be used with caution, since the entire TCL work file is rewritten every time this command is used.

Examples:

```
RESEQ
RES 100-1000
RESEQ 100        % (reseq entire file, base = 100)
RES 1000 + 1000
RES + 500
RESEQ 100-2000 100 + 10
```

RMERGE

Syntax:



Semantics:

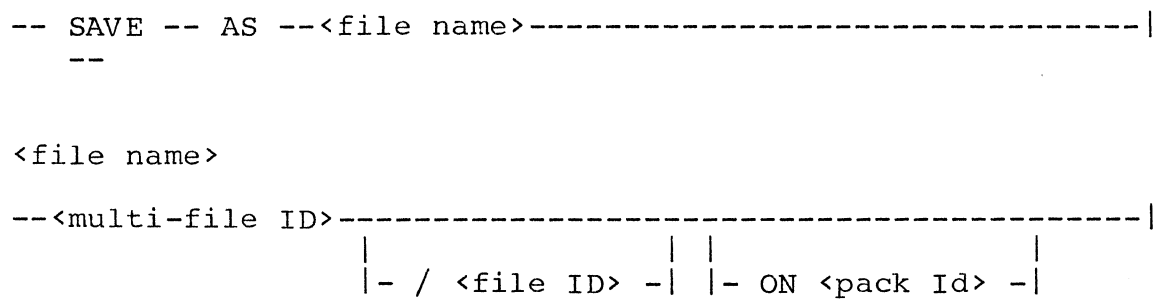
The RMERGE command merges all or part of the specified file name with the TCL work file by collating the sequence numbers. The merged records become part of the TCL work file. When records in the merged file and the TCL work file have identical sequence numbers, the mreged file records are retained, and the TCL work file records are ignored.

Examples:

```
RMERGE TEMPORARY/TCL
RM PERM/TCL.11.08  100-2000
```

SAVE

Syntax:



Semantics:

The SAVE command is used to save the contents of the TCL work file on disk. If the specified file name is already on disk, the TCL work file is not saved, and an error message is issued. The saved file can be merged with the primary TCL specifications file to incorporate new formats into the MCS on a permanent basis the next time the TCL compiler is run.

Examples:

```
SAVE AS USER1/NEWFORMATS
SA AS FORMPACK/TCLFORMATS/NOV.08.79
```

SEQUENCE

Syntax:

```
---<integer>----<TCL source image>-------------------------|
```

Semantics:

The SEQUENCE command allows insertion of source images directly into the user's TCL work file. The user enters a sequence number, starting at the first position on the terminal, followed by any valid TCL format or function declaration. The first eight positions on the terminal screen are reserved for the sequence number.

However, leading zeros may be suppressed, since the Format Generator terminates a sequence number upon receipt of the first non-integer character or after eight continuous integer characters. To enter an integer character immediately after the sequence number, the sequence number must occupy all eight sequence number positions. When a source image with the same sequence number already exists in the TCL work file, the work file image is replaced by the new input.

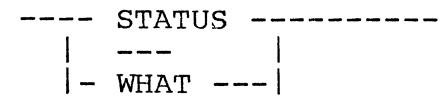
This command can be entered at any time during the session, except when a terminal is in an Expecting Format Layout mode. It can be used to modify a format that was retrieved using the GET command, to modify a format generated from translation of a format layout, or to create a format using the TCL syntax.

Examples:

```
00000100FORMAT F1 ("THIS IS A FORMAT",A6,I2,4(A8,X3)).
500FORMAT F2 ("ANOTHER FORMAT",I5,3(A2),X3,"FINI").
1000   ,"THIS IS THE MIDDLE OF A FORMAT",A6,I3,X2.
00300FUNCTION BOOLEAN (1: "YES", 0: "NO").
```

STATUS

Syntax:



Semantics:

The STATUS command is used to obtain information about a particular station that is attached to Format Generator. The information obtained using the STATUS command includes:

- Station RSN.
- Current terminal attributes.
- Current TCL work file size.
- Debug option status (set or reset).
- Name of format currently in the Format Layout file.

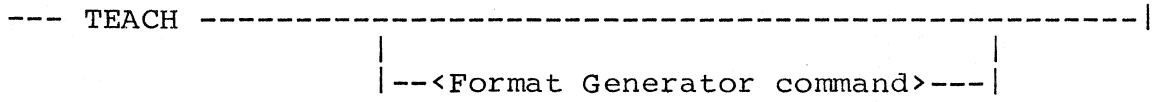
If a STATUS command is entered while the terminal is in an Expecting Format Layout mode, the status report includes the name of the new format to be entered into the Format Layout file and displays the message Expecting Format Layout, indicating that a format layout is expected. While in this mode, the STATUS command must be entered starting at the first position on the terminal, and STATUS must be the only word entered. The STATUS and WHAT commands are synonymous.

Example:

```
STATUS
```

TEACH

Syntax:



Semantics:

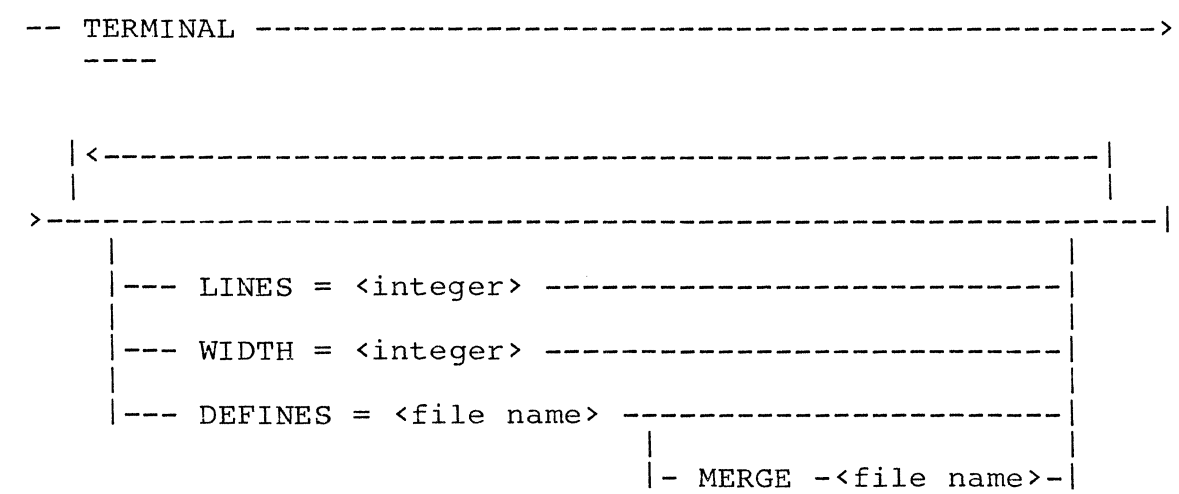
The TEACH command is used to obtain a list of valid Format Generator commands or to learn the syntax of a particular command. TEACH entered alone displays a one-screen listing of all valid Format Generator commands and their abbreviations. Also included is the syntax of <file name> and <sequence range>, which are syntax elements used in a number of commands. The word TEACH accompanied by any Format Generator command displays the syntax for that particular command with a short explanation of its purpose.

Examples:

```
TEACH
TEACH COMPILE
```

TERMINAL

Syntax:



Semantics:

The TERMINAL command allows the user to set certain attributes that may vary from terminal to terminal. The lines and width attributes affect the size of messages generated by the Format Generator, as well as how format layouts are broken up. If incorrect lines and/or width attributes are entered, output to the terminal becomes garbled.

The Defines attribute specifies the name of the current Defines file for the user as described under the DEFINE command. If the Defines file does not exist in the disk directory, a new file (empty) is created for the user. This file becomes the current Defines file. If a Merge file name is specified and exists, the defines from this file are merged into the Defines file. If the same define(s) exist in both files, the Merge file defines are used. Any previous Defines file is saved on disk.

TERM entered alone displays the current attributes. TERM entered with an attribute is used to set the attributes and to display the new terminal attributes. The default values are:

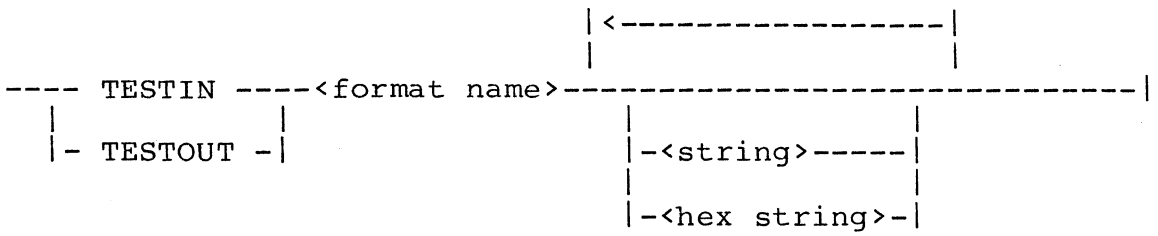
```
Line    = 24
Width   = 80
Defines = GEMCOS/DEFINES.
```

Examples:

```
TERMINAL LINES=12 WIDTH=40
TERM DEFINES = FORMGEN/DEFINES MERGE USER1/DEFINES
TERMINAL
```

TEST

Syntax:



Semantics:

The TEST command allows the user to apply a format to a test message and to return the resulting formatted message to the terminal. The format must be a successfully compiled format created during the current formatting session. It may be a format created directly in the TCL work file or indirectly using a format layout.

If no test message is sent, the format is applied to a blank message. Characters that do not exist on the terminal's keyboard may be entered in the form of hex strings.

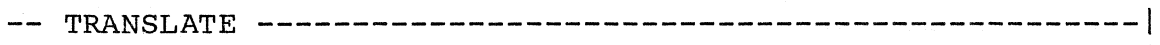
Some GEMCOS format editing phrases operate differently depending on whether the format is executed as an input format or an output format. TESTIN and TESTOUT execute a format as an input or output format respectively.

Examples:

```
TESTIN F1 "ABCD" "XYZ" 4"8182" "1234"
TESTOUT F2 " X " 4"85" " Y "
TESTOUT F3
```

TRANSLATE

Syntax:



Semantics:

The TRANSLATE command is used to translate the format layout into TCL source images and to put these images into the TCL work file. The translated format is placed at the end of the TCL work file. If the format already exists in the work file, the old TCL source images are removed.

If control information has been specified for a line of the format, and the line contains define names, these defines are expanded into their component parts, and a comma is inserted between parts before translation is performed. If error 69 or 97 occurs, a line number is supplied as part of the error message. (See Appendix B for text of error messages.) The LINE command can then be used to determine the nature of the error.

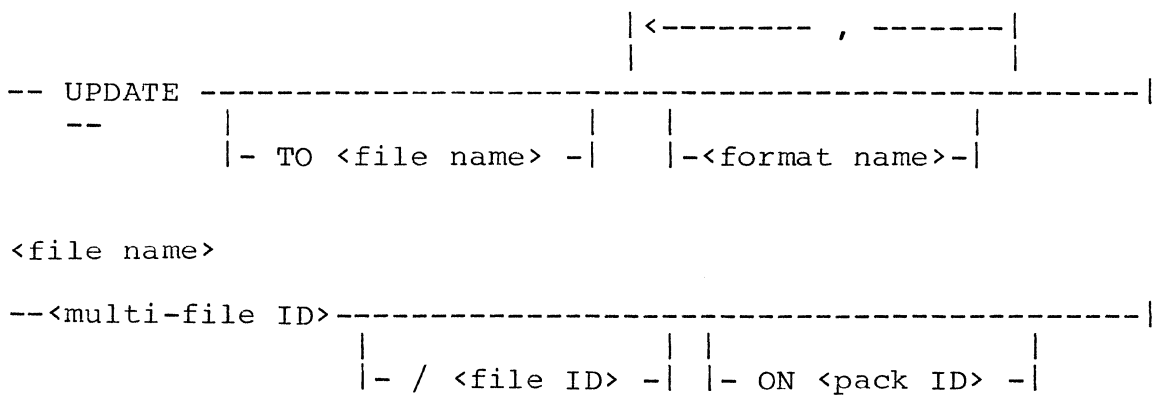
When AUTOCOMPILE is set to TRUE, the TRANSLATE command is automatically executed upon receipt of a format layout.

Example:

```
TRANSLATE
```

UPDATE

Syntax:



Semantics:

The UPDATE command is used to place an updated version of existing formats into an object Format file. An error results if the specified formats do not already exist in the object Format file or if the format file does not exist. This command is NOT valid if the Format Generator is NOT executed in UPDATE mode. UPDATE mode is the system default (SW 0 = 0).

If no file name is specified, the current live GEMCOS format file, named GEMCOS/MCSFORMATS, is assumed. Otherwise, the specified file is updated. If no file name or format name is specified, all formats in the user's object Test Format file are eligible for update. Otherwise, only the specified formats are updated.

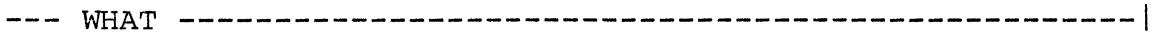
This command does not actually cause the updated version of a format to be made available to stations in a live GEMCOS environment. Rather, it places an updated copy of the format in the specified Format file. For these updated formats to be made available to stations in the live GEMCOS environment, a *UPD network control command (NCC) must be performed through a GEMCOS control station or the ODT (On-line Data Terminal).

Examples:

```
UPDATE
UP TO TEST/FORMATFILE
UP TO USER1/FORMFILE FMT1, FMT2, FMT3
```

WHAT

Syntax:



Semantics:

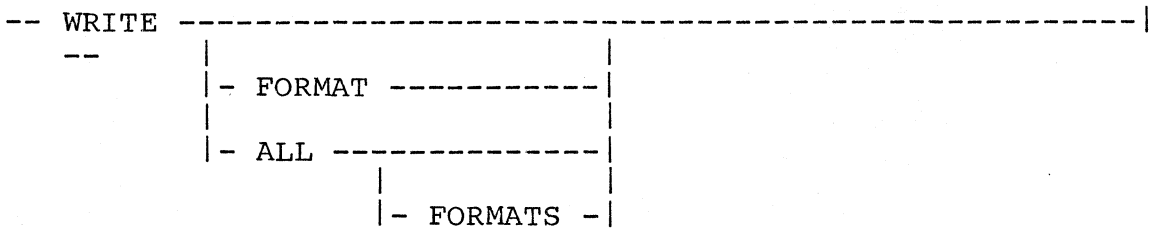
The WHAT command is used to obtain information about a particular station that is attached to Format Generator. Refer to the STATUS command for more detailed discussion. The STATUS and WHAT commands are synonymous.

Example:

```
WHAT
```

WRITE

Syntax:



Semantics:

The WRITE command, when entered alone, writes the contents of the TCL work file to the line printer. When the FORMAT option is used, the contents of the format layout work file is written to the line printer. Only the text and data portions of each line are printed.

Examples:

```
WRITE
WR FORMAT
```

APPENDIX A

FORMAT MAKER INFORMATION

The following three tables contain further B 1000 GEMCOS Format Maker information.

FORMAT MAKER FILES

The B 1000 GEMCOS Format Maker files are summarized in the following table.

Name	Device	Explanation
Internal: MCSREMOTE External: MCSREMOTE	Remote	A remote file opened by the Format Maker for I/O between the Format Maker and the user. Default number of stations is 10. Can be file equated.
Internal: MCSTESTFOR External: MCSNONFORM/t	Disk	The Test Format file in which t is the <time stamp>. If empty, purged at end of session. If contains formats, saved at end of session.
Internal: MCSGETFILE External: none	Disk	File created by the GET command. Closed at completion of GET.
Internal: MCSOPNFILE External: none	Disk	File created by the OPEN command. Input only. Closed at completion of OPEN.
Internal: MCSLIBRARY External: MCSLIBRARY/t	Disk	The Library file in which the t is the <time stamp> and which is used to store COBOL entries for each indicated format field.

Internal: MCSPRINT External: MCSPRINT	Printer	A Printer file used in WRITE. Default device setting is printer or printer backup. Output only.
Internal: MCSMONITOR External: MCSMONITOR	Printer	A Printer file used by DEBUG. Default device setting is printer or printer backup.

SYSTEM LIMITS AND DEFAULTS

General	Limit/Default
Format name:	1 to 6 characters
Maximum users:	1
UPDATE mode:	TRUE
Maximum fields in format:	144
Maximum number of formats:	804
Maximum size of Library file:	20000 records
Terminal size:	1920 bytes (TD compatible)

UPDATE Command

Number of characters of input:	Up to 1000
Number of named formats:	1 to 166
Maximum number of formats:	400 before *UPD FORMATS
UPDATE file default name:	GEMCOS/NONINTERS

Program Switches

SW0	If equal to 0, UPDATE cannot be used.
SW1	If equal to 1, DEBUG trace will be turned on at beginning-of-job.

FORMAT MAKER ERROR MESSAGES

ERROR 00: UNRECOGNIZED COMMAND

ERROR 01: ACCESS DENIED, MAX USERS EXCEEDED

ERROR 02: UNEXPECTED TOKEN

ERROR 03: <not used>

ERROR 04: MISSING FILE NAME

The command did not contain a necessary file name. Reenter the command with the file name.

ERROR 05: INVALID FILE NAME

The file name is not a valid file title. Reenter command with a valid file name.

ERROR 06: FILE MISSING - <file name>

ERROR 07: NOT ALLOWED TO USER CURRENT TESTFILE WITH COMMAND

ERROR 08: INCORRECT RELEASE LEVEL IN TESTFILE - <file name>

ERROR 09: FILE IS EMPTY - <file name>

ERROR 10: MAXIMUM NUMBER OF OUTPUT FIELDS EXCEEDED

ERROR 11: MAXIMUM SIZE OF COMPACTED RECORD EXCEEDED

ERROR 12: FORMAT NOT FOUND

ERROR 13: FILE ALREADY ON DISK - <file name>

ERROR 14: NO MORE ROOM IN LIVE FORMAT FILE

ERROR 15: NO CURRENT FORMAT

ERROR 16: MISSING FORMAT NAME

The command did not contain a necessary format name. Reenter command with a valid format name.

ERROR 17: INVALID FORMAT NAME

The format name entered is longer than six characters or contains an illegal character. Reenter the command with a valid format name.

ERROR 18: INVALID ATTRIBUTE SPECIFICATION

ERROR 19: ATTRIBUTE ALREADY SPECIFIED

ERROR 20: '=' EXPECTED

ERROR 21: MISSING 'FROM <FILE NAME>'

ERROR 22: MISSING ATTRIBUTE VALUE

ERROR 23: MISSING FIRST QUOTE

ERROR 24: MISSING TEST DATA

ERROR 25: MISSING FINAL QUOTE

ERROR 26: TOO MUCH TEST DATA

ERROR 27: NOT ENOUGH TEST DATA

ERROR 28: MISING DIRECTORY RECORD IN FILE

ERROR 29: NOT A VALID GEMCOS FORMAT FILE - <file name>

ERROR 30: ',' EXPECTED

ERROR 31: MAXIMUM NUMBER OF FORMATS EXCEEDED

The user has attempted to enter more than 804 formats in the format file. Delete unneeded formats before adding more or create a new format file.

ERROR 32: DUPLICATE FORMAT NAME

The user has attempted to create a format with an existing format name. Create format with unique, valid format name.

ERROR 33: FORMAT MAKER NOT RUN IN UPDATE MODE

ERROR 34: FORMAT WAS DELETED

ERROR 35: **FORMAT NAME MISMATCH IN TESTFILE**

ERROR 36: COMMAND MAY NOT BE USED AT THIS TIME

The IGNORE command has been transmitted when the terminal was not in Expecting Format Layout mode.

ERROR 37: UPDATE TABLE IN LIVE FORMAT FILE FULL

ERROR 38: EOF/EXCEPTION ERROR

ERROR 39: FORMAT NOT FOUND - NO FORMATS UPDATED

APPENDIX B

FORMAT GENERATOR INFORMATION

The following three tables contain further B 1000 Format Generator information.

FORMAT GENERATOR FILES

The B 1000 Format Generator files are summarized in the following table.

Name	Device	Explanation
Internal: MCSREMOTE External: MCSREMOTE	Remote	A remote file opened by the Format Generator for I/O between the Format Generator and the user. Maximum number of stations per copy is 1. Can be file equated.
Internal: MCSTCLWRKn External: MCSTCLWRKN/t	Disk	A TCL workfile where n is an integer from 0 to 9 indicating the the user and t is the <time stamp>. Purged at the end of the formatting session.
Internal: MCSLAYOUTn External: MCSLAYOUTn/t	Disk	The Format Layout file in which n is an integer from 0 to 9, indicating the user, and t is the <time stamp>. t is the same for all users. Purged at end of session.
Internal: MCSLIVEFOR External: GEMCOS/MCSFORMATS	Disk	Used to read and write formats from any live GEMCOS format file.
Internal: MCSTESTFOR External: MCSTSTFORn/t	Disk	The Test Format file in which n is an integer from 0 to 9 indicating the user and t is the <time stamp>. t is the same for all users.

		Purged at the end of the session.
Internal: MCSDEFINE External: GEMCOS/DEFINES	Disk	The Defines file. System-supplied or created by user. User-created Defines files are saved at the end of a session.
Internal: MCSUTILITY External: none	Disk	System Utility file used in RESEQ and other system functions.
Internal: MCSMERGE External: none	Disk	System Utility file used in MERGE and RMERGE. Not accessible by user.
Internal: MCSPRINT External: MCSPRINT	Printer	A Printer file used in WRITE. Default device setting is printer or printer backup. Output only.
Internal: MCSMONITOR External: MCSMONITOR	Printer	A Printer file used by DEBUG. Default device setting is printer or printer backup.

SYSTEM LIMITS AND DEFAULTS

General	Limit/Default
Format name:	1 to 17 characters
Define name:	1 to 17 characters
Sequence number:	0 thru 99999999
Maximum users:	10
UPDATE mode:	TRUE
Maximum file size:	65535 records

COMPILE Command

Number of characters of input:	Up to 1000
Number of named formats:	1 to 58
UPDATE file default name:	GEMCOS/MCSFORMATS

DEFINE Command

Number of occurrences of <define type>:	1 to 255
Length of string or hex string, including quotes:	Up to 173 characters
<Define loc> <integer>:	1 to 65535
Default Defines file name:	GEMCOS/DEFINES
Define nesting:	Up to 9

DELETE Command

Default delete range:	Entire TCL workfile

FORMAT Command

Default <format attributes> value:	FALSE

GET Command

Default format file name:	User's Test Format file

LINE Command	Limit/Default
Number of characters input:	Up to 1000
Length of <control specs>:	Up to 173 characters

LIST Command

Default list range:	Entire TCL workfile

MERGE Command

Merge file record size:	80 or 90 characters

RESEQ Command

Default resequence range:	Entire TCL workfile starting at 100 With increment of 100

RMERGE Command

RMERGE file record size:	80 or 90 characters

TERMINAL Command

Number of lines:	1 to 24; default is 24
Number of characters per line:	1 to 80; default is 80

TEST Command

Length of input data:	Up to 600 characters
Length of output data:	Up to 2000 characters
Forms request:	If <test message> is empty, no format clause can exceed 600 characters

TRANSLATE Command

(In conjunction with LINE command)

Length of <control string>:	Up to 1520 characters
Length of <control specs>:	Up to 68 characters without a comma

UPDATE Command

Number of characters of input:	Up to 1000
Number of named formats:	1 to 58
UPDATE file default name:	GEMCOS/MCSFORMATS

FORMAT GENERATOR ERROR MESSAGES

ERROR 00: UNRECOGNIZED COMMAND

ERROR 01: ACCESS DENIED, MAX USERS EXCEEDED

ERROR 02: UNEXPECTED TOKEN

ERROR 03: INTEGER EXPECTED

ERROR 04: MISSING FILE NAME
The command did not contain a necessary file name. Re-enter the command with the file name.

ERROR 05: INVALID FILE NAME

The file name is not a valid file title. Re-enter the command with a vaid file name.

ERROR 06: FILE MISSING - <file name>

ERROR 07: FILE LOCKED - <file name>

ERROR 08: FILE IS NOT FORMAT GENERATOR COMPATIBLE - <file name>

ERROR 09: FILE IS EMPTY - <file name>

ERROR 10: TCL WORKFILE IS EMPTY

ERROR 11: BEGINNING SEQUENCE NUMBER EXCEEDS ENDING SEQUENCE NUMBER

ERROR 12: 'AS' EXPECTED

ERROR 13: FILE ALREADY ON DISK - <file name>

ERROR 14: SEQUENCE NUMBERS HAVE EXCEEDED MAXIMUM SIZE

ERROR 15: RESEQUENCED RECORDS HAVE OVERLAPPED EXISTING RECORDS

ERROR 16: MISSING FORMAT NAME

The command did not contain a necessary format name. Re-enter the command with a valid format name.

ERROR 17: INVALID FORMAT NAME

The format name is longer than six characters or contains an illegal character. Re-enter the command with a valid format name.

ERROR 18: INVALID ATTRIBUTE SPECIFICATION

ERROR 19: ATTRIBUTE ALREADY SPECIFIED

ERROR 20: '=' EXPECTED

ERROR 21: COMMAND MAY NOT BE USED AT THIS TIME

The IGNORE command was transmitted when the terminal was not in Expecting Format Layout mode.

ERROR 22: LINE NUMBER EXPECTED ON FIRST LINE

ERROR 23: LINE NUMBER DOES NOT EXIST

ERROR 24: INVALID CHARACTER(S) ON DATA LINE

ERROR 25: INVALID CHARACTER(S) ON LOCATION LINE

ERROR 26: UNBALANCED LOCATION & CONTROL INFORMATION

ERROR 27: TEXT & DATA CHARACTER OVERLAP

ERROR 28: FORMAT LAYOUT FILE IS EMPTY

ERROR 29: FILE IS NOT A VALID GEMCOS FORMAT FILE - <file name>

ERROR 30: ',' EXPECTED

ERROR 31: MAXIMUM NUMBER OF FORMATS EXCEEDED

The user has attempted to enter more than 804 formats in the format file. Delete unneeded formats before adding more or create a new format file.

ERROR 32: DUPLICATE FORMAT NAME

The user has attempted to create a format with an existing format name. Create format with a unique, valid format name.

ERROR 33: FORMAT GENERATOR NOT RUN IN UPDATE MODE

ERROR 34: EOF ON WORKFILE - NOT ALL FORMATS COMPILED

No more room remains in the work file to accommodate compiled formats. Delete unneeded formats before attempting to compile any more or create a new format file.

ERROR 35: '.' OR ',' EXPECTED

ERROR 36: '(' EXPECTED

ERROR 37: ')' EXPECTED

ERROR 38: COMMA, CLOSING QUOTE OR ')' EXPECTED

ERROR 39: ']' EXPECTED

ERROR 40: RESIDENT EXPECTED

ERROR 41: ';' OR ',' EXPECTED

ERROR 42: 'FOR' EXPECTED

ERROR 43: MISSING MAX REPETITIONS IN REPEAT PART

The user has attempted to merge a hand-written TCL specification into a Format Generator TCL file. The specification lacks a number indicating the number of times an operation is to be repeated. Supply a number from 1 to 255.

ERROR 44: 'OR' EXPECTED

ERROR 45: MISSING INTERNAL SIZE

ERROR 46: MISSING FIELD WIDTH

ERROR 47: INVALID OR MISSING VARIABLE DECLARATION FIELD LENGTH

ERROR 48: INVALID OR MISSING VARIABLE IDENTIFIER

ERROR 49: MORE THAN 32 LEVELS OF PARENTHESES

ERROR 50: INVALID REPEAT PART VALUE OR INVALID ITEM PHRASE

ERROR 51: INVALID MAX REPETITIONS IN REPEAT PART

ERROR 52: UNDECLARED VARIABLE IDENTIFIER

ERROR 53: INVALID INTERNAL SIZE

ERROR 54: INVALID OR MISSING FIELD DELIMITER

ERROR 55: INVALID FIELD WIDTH SIZE

ERROR 56: INVALID OR MISSING SKIP DELIMITER

ERROR 57: MISPLACED REPEAT PART

ERROR 58: INVALID LOCATION SPECIFIER VALUE

ERROR 59: INVALID OR MISSING LOCATION SPECIFIER VALUE

ERROR 60: UNDECLARED FUNCTION IDENTIFIER

ERROR 61: MISSING FUNCTION IDENTIFIER

ERROR 62: INVALID OR MISSING ITEM TYPE

ERROR 63: INVALID OR MISSING INTERNAL SIZE

ERROR 64: TRANSLATE STRING CANNOT EXCEED 6 CHARACTERS

ERROR 65: INCOMPLETE FORMAT STATEMENT

ERROR 66: DUPLICATE VARIABLE DECLARATION

ERROR 67: UPDATE TERMINATED - NOT ALL NAMED FORMATS FOUND

The user has attempted to update a nonexisting format name. Check list of UPDATE messages for the last entry. Re-enter UPDATE command with the correct format name.

ERROR 68: FORMAT NOT FOUND IN FORMAT FILE

A nonexisting format name was entered. Enter the correct format name.

ERROR 69: <control specs> TOO LARGE FOR TCL WORKFILE

ERROR 70: INVALID VALUE FOR THIS ATTRIBUTE

ERROR 71: NON-NUMERIC VALUE IN SEQUENCE FIELD OF MERGE RECORD

ERROR 72: FILE IS NOT A VALID GEMCOS DEFINES FILE - <file name>

ERROR 73: MISSING DEFINE NAME

ERROR 74: INVALID DEFINE NAME

ERROR 75: DEFINE NOT FOUND IN DEFINES FILE

ERROR 76: INVALID INTEGER VALUE

ERROR 77: MISSING TOKEN

ERROR 78: STRING INVALID, UNBALANCED OR TOO LONG

ERROR 79: STRING EXPECTED

ERROR 80: TEST DATA TOO LARGE

A data string was entered with the TESTIN or TESTOUT command, but the string was too large for the format being tested. Re-enter the command with a shorter data string.

ERROR 81: TEST ERR, DEST PTR OUT OF BOUNDS

ERROR 82: TEST ERR, SOURCE PTR OUT OF BOUNDS

ERROR 83: TEST ERR, NON-DIGIT IN INTEGER FIELD

ERROR 84: TEST ERR, MISSING SKIP DELIMITER

ERROR 85: TEST ERR, VARIABLE REPEAT ON INPUT

ERROR 86: TEST ERR, MISSING DELIMITER

ERROR 87: TEST ERR, INVALID TRANSLATE FIELD

ERROR 88: INVALID FUNCTION IDENTIFIER

ERROR 89: ':' EXPECTED

ERROR 90: MISSING KEYWORD 'EXTERNAL'

ERROR 91: MISSING KEYWORD 'INTERNAL'

ERROR 92: MISSING TRANSLATION LIST

ERROR 93: INVALID FUNCTION TYPE

ERROR 94: EXTERNAL STRING LENGTH MUST BE 1 TO 6 CHARACTERS

ERROR 95: INTERNAL STRING LENGTH MUST BE 1 TO 6 CHARACTERS

ERROR 96: INCOMPLETE FUNCTION STATEMENT

ERROR 97: <control string> TOO LARGE TO TRANSLATE

ERROR 98: DEFINE NESTING LIMIT EXCEEDED

ERROR 99: SCREENSIZE LIMIT EXCEEDED - COMMAND NOT ACCEPTED

A command has been entered that exceeds the screensize limit defined using the TERMINAL command.

INDEX