Eloquence Forms Manual
B.06.32

Edition E1202

Legal Notices

Restricted Rights Legend

Acknowledgments

Marxmeier Software AG
Besenbruchstrasse 9
42285 Wuppertal
Germany

Printing History

The manual printing date indicates its current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. New editions are complete revisions of the manual.The dates on the title page change only when a new edition or a new update is published.

Manual updates may be issued between editions to correct errors or document product changes. Manuals that are published on the Eloquence website (www.hp-eloquence.com/doc) may be updated more often, please visit this website periodically for the most recent versions. To ensure that you receive the updated or new editions, you should also subscribe to the appropriate product support service.

The software code printed alongside the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

First Edition	Apr 1990	A.01.00
Second Edition	July 1991	A.03.00
Third Edition	January 1997	A.06.00
Fourth Edition	October 1997	A.06.00
Fifth Edition (E1202)	December 2002	B.06.32

Printed in the Federal Republic of Germany.

Printing History

Table of Contents

Contents

1

Introduction

The Eloquence Forms software gives you a means to draw a form image on the display screen. You can also specify input and output fields and the order in which these fields are to be accessed by a user and a program. Once a program has been written that uses a form, that form can be modified without having to modify the program.

The process of creating forms is described in page 13 . Modifying a form is described in page 27 . The information about the form that a program uses is described in page 31 . The form control statements are described in page 41 .

NOTE: Eloquence Forms software requires a character I/O interface and so it is not supported on the NT platform.

Conventions

The statements in this manual use the same syntax conventions as in the *Eloquence Manual*.

- **Bold type** is used when a new term is introduced.
- `Computer font` indicates text to be input exactly as shown or text that is output from the system.
- *Italic type* is used for emphasis and titles of publications. It is also used to indicate parameters that are user defined.
- KEYCAP represents a key on the keyboard.
- Shading represents the softkeys displayed on the computer screen.
- …indicates that the previous variable can be repeated.
- [ ] indicates that information inside the brackets is optional. If there are brackets within brackets, the information within the inner bracket may only be specified if the information in the outer bracket is specified. Information may also be stacked in brackets. For example, A or B or neither may be selected when the following is shown:

$$\begin{bmatrix} A \\ B \end{bmatrix}$$

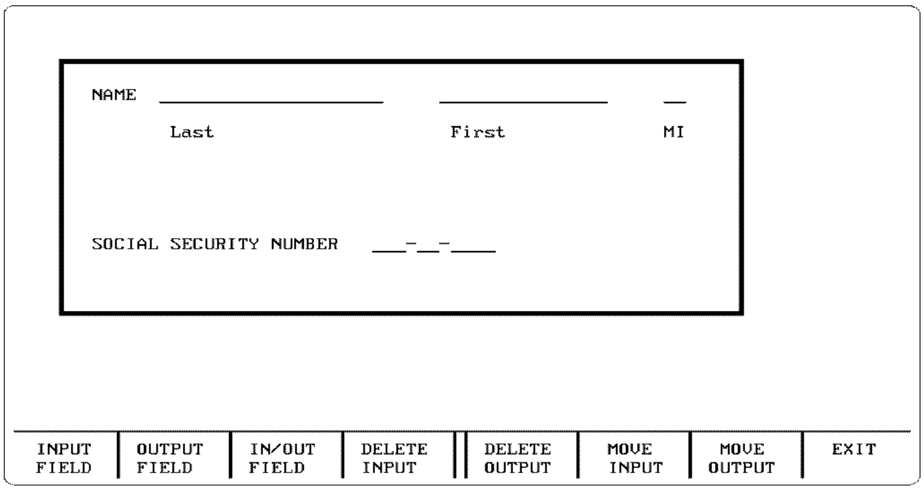
- { } indicates that one of the choices stacked within the braces must be selected. For example, A or B or C must be selected when the following is shown:

$$\begin{Bmatrix} A \\ B \\ C \end{Bmatrix}$$

NOTE: Notes contain important information that is set off from the text.

Forms Introduction

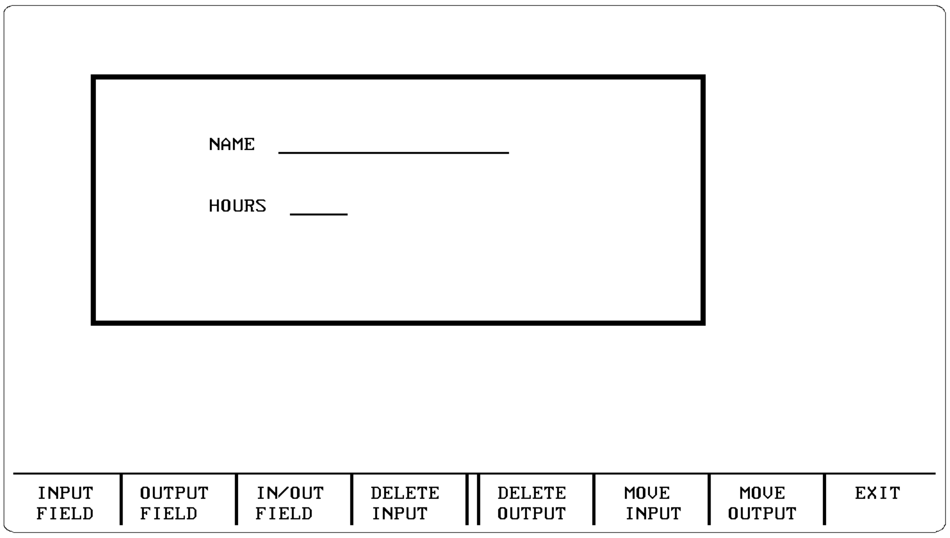
A **form** is an image that is displayed, providing input and output formats for program use. The user then enters the correct information into each specified blank space. Instead of prompting for each item, the program can display the form. For example:



```
NAME  ________________________    __________________    __
        Last                           First              MI


SOCIAL SECURITY NUMBER   ___-__-____
```

The user fills in the blanks, pressing either RETURN or TAB after each item. (The way in which the program is written will determine which of the two keys can be used. Refer to page 31). The cursor then moves automatically to the next item to be input.

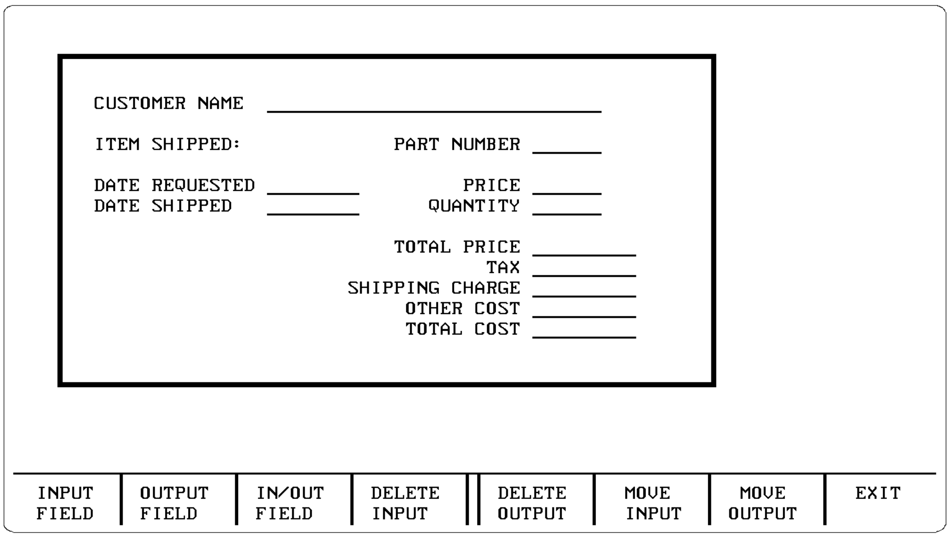
A program can also output information to a form. Assume a program requests the number of hours an employee worked in a week. A form for this might look like the following:



```
NAME  ____________________

HOURS _____
```

where the program fills in the name and the user fills in the hours.

The two previous examples show very simple forms; it may have been easier to use simple input and output statements. However, if more information about a topic is wanted, the form becomes the easier method. For example, assume a program keeps track of customer shipments. One form it uses could look as follows:



```
CUSTOMER NAME  ______________________________

ITEM SHIPPED:                PART NUMBER ______

DATE REQUESTED ________            PRICE ______
DATE SHIPPED   ________         QUANTITY ______

                             TOTAL PRICE _________
                                     TAX _________
                         SHIPPING CHARGE _________
                              OTHER COST _________
                              TOTAL COST _________
```

The program can display this form and prompt the user to fill it in. The program can read entries, then blank out the form and begin again.

A form can be created to look like a printed form currently in use. In this way, a user can easily type the information from a paper to an identical form displayed on a terminal.

2

Creating Forms

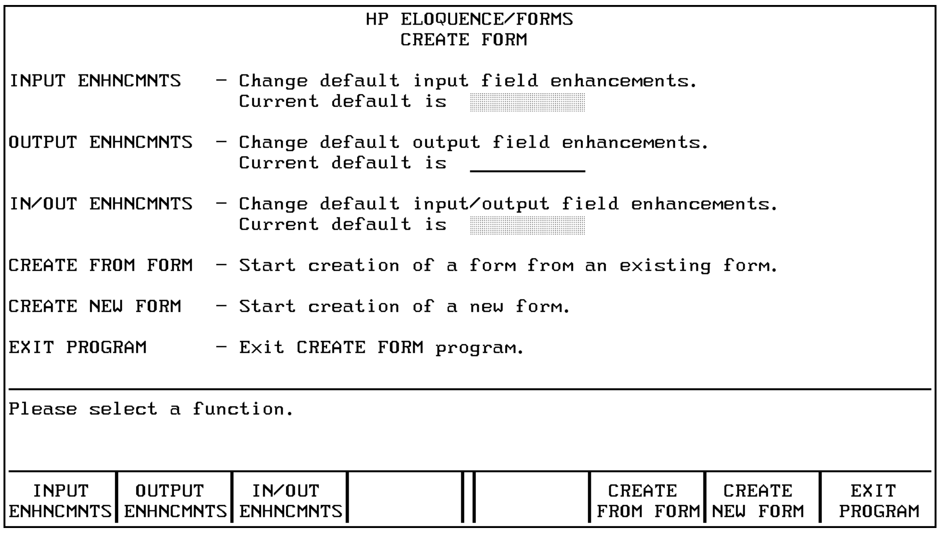
To create a form, use the Create Form (CFORM) program. Start Eloquence and type the following:

```
RUN”CFORM[,volume label]” RETURN
```

After you execute the RUN command, the display shows you the definition of each of the softkeys on the display and shows a short explanation of each one. This is called the initial CREATE FORM menu.

NOTE: For the CREATE FORM menu to appear, a VOLUME statement pointing to the directory /opt/eloquence/share/prog must be included in the global configuration file eloq.config. The sample global configuration file d.eloq.config contains such a statement (**`VOLUME SYSTEM :Z2,7,0 /opt/eloquence/share/prog`**).

Initially the softkeys are defined as follows:



```
                              HP ELOQUENCE/FORMS
                                 CREATE FORM

INPUT ENHNCMNTS    - Change default input field enhancements.
                     Current default is  ▒▒▒▒▒▒▒▒▒▒

OUTPUT ENHNCMNTS   - Change default output field enhancements.
                     Current default is  __________

IN/OUT ENHNCMNTS   - Change default input/output field enhancements.
                     Current default is  ▒▒▒▒▒▒▒▒▒▒

CREATE FROM FORM   - Start creation of a form from an existing form.

CREATE NEW FORM    - Start creation of a new form.

EXIT PROGRAM       - Exit CREATE FORM program.

Please select a function.

  INPUT  | OUTPUT  | IN/OUT  |         ||         | CREATE  | CREATE  |  EXIT
ENHNCMNTS|ENHNCMNTS|ENHNCMNTS|         ||         |FROM FORM|NEW FORM | PROGRAM
```

INPUT ENHNCMNTS — The input enhancements softkey. When an area (or field) is to be used exclusively for input, you can set 1) the fill character for that field [a fill character is the character that is repeated in the field during the creation of the form] and 2) the way in which the character is displayed [inverse video, underline, etc.].

OUTPUT ENHNCMNTS — The output enhancements softkey. A field can be defined exclusively for output. This field can be visually enhanced in the same manner as an input field.

IN/OUT ENHNCMNTS — The input and output enhancement softkey. A field can also be defined to accept both input and output. This field can be visually enhanced in the same manner as an input or an output field.

CREATE FROM FORM — Allows you to specify a form that already exists to be used as a basis for creating a new form. When this key is pressed, a new menu is displayed. See the sub-section titled page 17 .

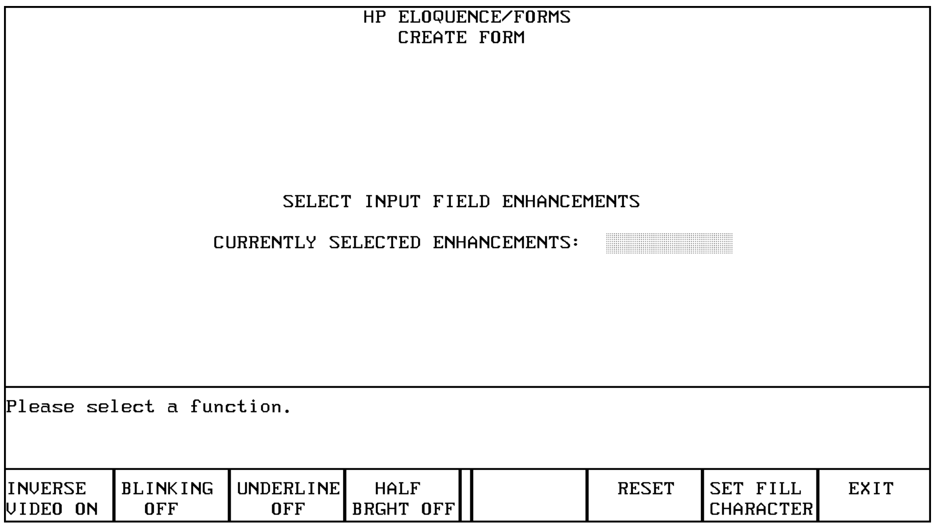
CREATE NEW FORM — Clears the display so you can create a new form. A new menu is displayed.

EXIT PROGRAM — Terminates the program.

Display Enhancements

When CFORM is run, a short explanation of each softkey is displayed. In addition, the current enhancements for input, output and input/output fields are shown. To change one of the field enhancements, press the related softkey.

For example, assume that the enhancement for input fields is inverse video and the fill character is a blank. To change the enhancements or fill character, you press the INPUT ENHNCMNTS softkey. The following information is shown on your display:



```
                         HP ELOQUENCE/FORMS
                            CREATE FORM


                   SELECT INPUT FIELD ENHANCEMENTS

             CURRENTLY SELECTED ENHANCEMENTS:

Please select a function.

INVERSE   BLINKING  UNDERLINE   HALF              RESET   SET FILL   EXIT
VIDEO ON    OFF        OFF    BRGHT OFF                   CHARACTER
```

SET FILL CHARACTER — Allows you to change the fill character. A fill character is used to indicate fields while the form is being created. The character is replaced by blanks when the form is stored. You will be asked to enter the fill character. The character will appear in the field next to CURRENTLY SELECTED ENHANCEMENTS after you press RETURN. You may change this character as many times as you wish by pressing this softkey again and entering a new character.

INVERSE VIDEO, BLINKING, UNDERLINE and HALF BRGHT — Toggled on and off. The on or off state is displayed. In this example INVERSE VIDEO is on, BLINKING is off, UNDERLINE is off, and HALF BRGHT is off.

RESET — Changes the enhancements and fill character to the value they had when the input, output, or input/output enhancement key was pressed.

EXIT — When you are satisfied with the selected enhancement, press the EXIT softkey to return to the initial menu.

Create Form

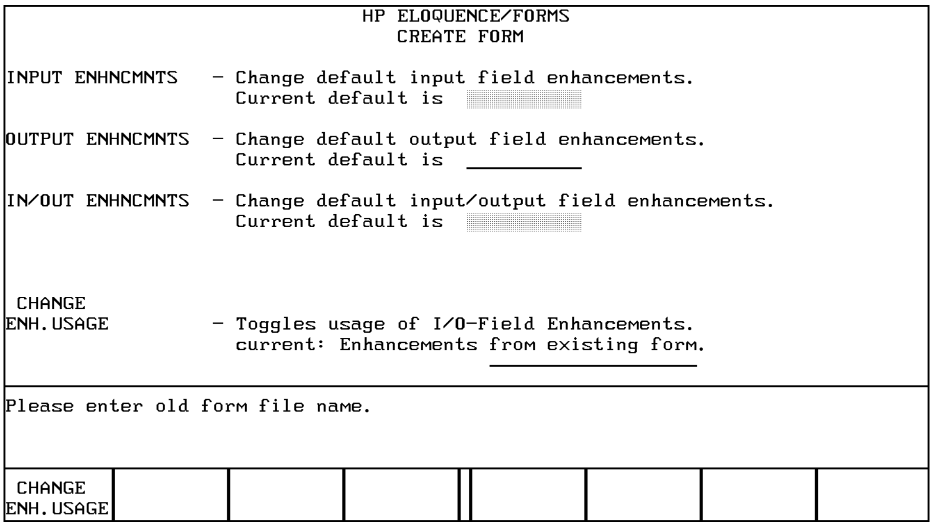
After you have selected the field enhancements, you are ready to create the form. There are two keys you can use to begin this process.

CREATE FROM FORM — Allows you to specify a form file name that already exists. (The file name may contain a device specifier or a volume label). The old form can be used as a basis for the new form. The old form will be displayed and a new menu will be set.

CREATE NEW FORM — Clears the display so you can create a completely new form. (Refer to the main forms-definition menu in page 17 .)

Creating a Form from an Existing Form

Press the CREATE FROM FORM softkey. The following screen appears:

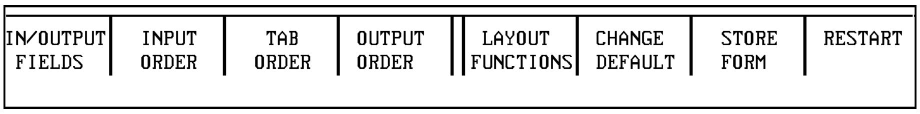


```
                           HP ELOQUENCE/FORMS
                              CREATE FORM

INPUT ENHNCMNTS    - Change default input field enhancements.
                     Current default is

OUTPUT ENHNCMNTS   - Change default output field enhancements.
                     Current default is

IN/OUT ENHNCMNTS   - Change default input/output field enhancements.
                     Current default is


 CHANGE
ENH.USAGE          - Toggles usage of I/O-Field Enhancements.
                     current: Enhancements from existing form.

Please enter old form file name.

 CHANGE
ENH.USAGE
```

CHANGE ENH. USAGE — Allows you to select the I/O-field enhancements from two sources:

- The original definition of the form you are modifying.
- The definitions selected previously. The current selection is indicated in the upper part of the screen.

Enter the file name of the existing form and press RETURN. The main forms-definition menu will appear, showing the following softkeys:



IN/OUTPUT FIELDS — Defines the location and length of input and output fields.

INPUT ORDER — Specifies the order in which the program reads the input fields.

TAB ORDER — Sets the order in which the user accesses each input field.

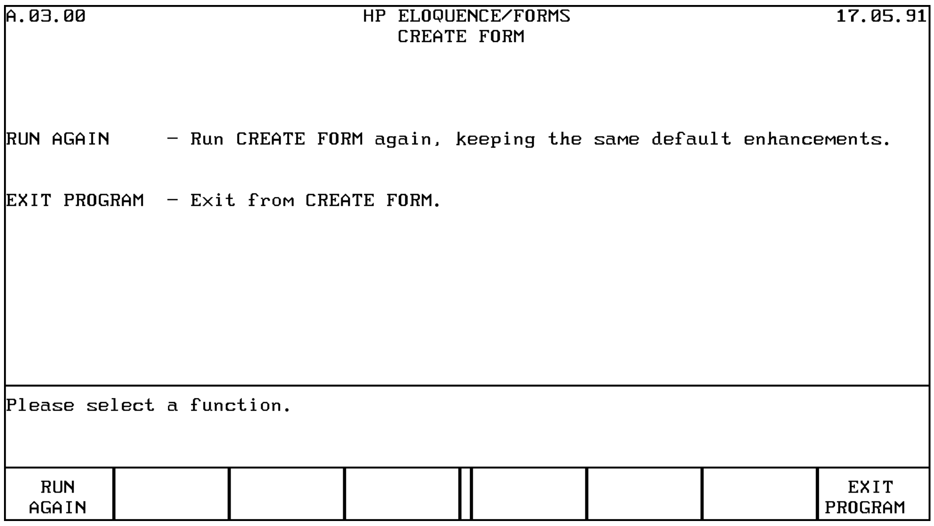
OUTPUT ORDER — Specifies the order in which the program accesses each output field.

LAYOUT FUNCTIONS — Used with the keyboard, it allows you to draw images (which are not fields) on the screen and use line-draw characters.

CHANGE DEFAULT — Allows you to change the current set of default enhancements. See the sub-section titled page 19 .

STORE FORM — Stores the form on a disk. The file name specified can be a maximum of nine characters. The extension .FORM is automatically appended to the specified file name.

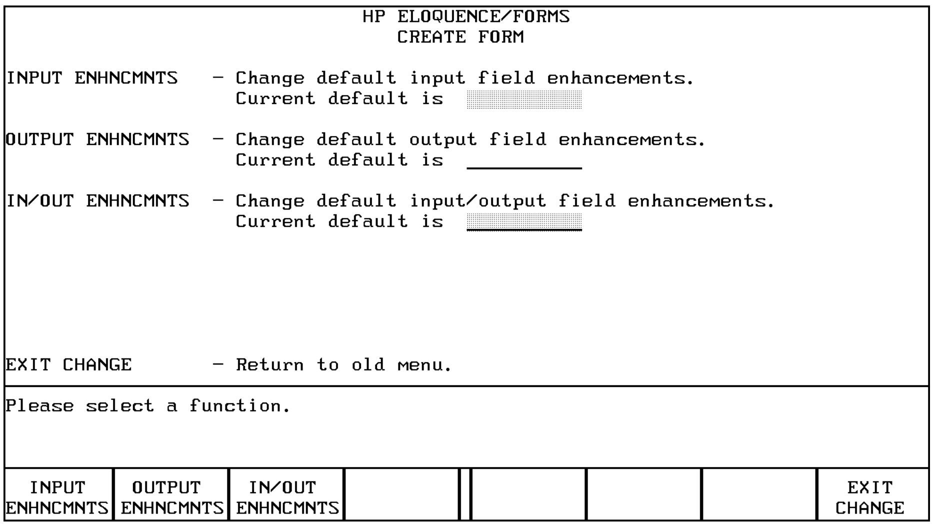
RESTART — Generates screen below.



```
A.03.00                        HP ELOQUENCE/FORMS                       17.05.91
                                  CREATE FORM


RUN AGAIN     - Run CREATE FORM again, keeping the same default enhancements.

EXIT PROGRAM  - Exit from CREATE FORM.








Please select a function.
```

Changing the Default Enhancements

When you press the CHANGE DEFAULT softkey, the following screen appears:



```
                              HP ELOQUENCE/FORMS
                                 CREATE FORM

INPUT ENHNCMNTS   - Change default input field enhancements.
                    Current default is

OUTPUT ENHNCMNTS  - Change default output field enhancements.
                    Current default is  __________

IN/OUT ENHNCMNTS  - Change default input/output field enhancements.
                    Current default is




EXIT CHANGE       - Return to old menu.

Please select a function.

  INPUT    OUTPUT    IN/OUT                                          EXIT
ENHNCMNTS ENHNCMNTS ENHNCMNTS                                       CHANGE
```

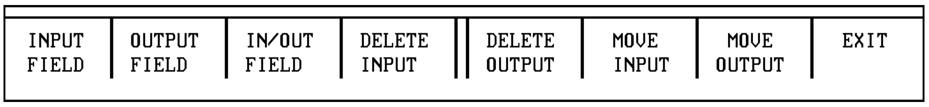
The default enhancements of input fields, output fields and input/output fields are changed by pressing the INPUT ENHNCMNTS, OUTPUT ENHNCMNTS or IN/OUT ENHNCMNTS softkeys respectively. This action takes you into the display enhancement selection screen. You can then proceed as described in the section titled page 15 (earlier in this section).

Input and Output Fields

The input and output fields are the means by which an application reads and writes to a form. These fields are defined by the following:

- The number and location of input and output fields.
- The order in which the input and output fields are accessed by the program and the user.

Use the IN/OUTPUT FIELD softkey to define the number and length of the input and output fields. When IN/OUTPUT FIELD is pressed, the following menu appears:



```
 INPUT   OUTPUT   IN/OUT   DELETE    DELETE   MOVE    MOVE    EXIT
 FIELD   FIELD    FIELD    INPUT     OUTPUT   INPUT   OUTPUT
```

INPUT FIELD — Defines an area (a field) that can only be used for input. You can define a maximum of 200 input fields for each form. To define an input field, position the cursor at the first character position and press INPUT FIELD. If the field is to be ten characters long, for example, press this softkey ten times. The input field enhancements and fill character are added automatically. (You defined the field enhancements and fill character before creating the form image.) A line may contain multiple input fields; however, input fields on the same line must be separated by at least one non-input field character.

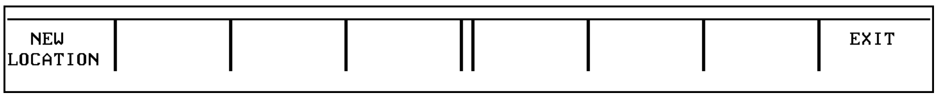
OUTPUT FIELD — Defines a field that can only be used for output. You can define a maximum of 200 output fields. Define each output field in the same manner as an input field.

IN/OUT FIELD — Defines a field that can be used for either input or output. Define each input/output field in the same manner as an input field or output field. Input/output fields must be separated from both input and output fields by at least one non-field character.

DELETE INPUT — To delete a field previously defined as an input field, you position the cursor on the left of the field to be deleted and press DELETE INPUT. If the field is an input/output field, only the input portion is deleted. That is, the input/output field becomes an output-only field. You delete everything to the right of the cursor, so you can use this facility to shorten field.

DELETE OUTPUT — Performs the same function as the DELETE INPUT softkey, and in the same way, except DELETE OUTPUT works on output fields.

MOVE INPUT — Moves an input field from one area of the form to another. First, position the cursor within the input field and press MOVE INPUT. A new menu appears:



Move the cursor to the location where the first character of the field is to be. Then press NEW LOCATION. The input field will be moved to the new location. The length of the field does not change. If the relocation is accomplished, with no errors, an EXIT is performed automatically. If the new location of the field would cause an overlap with another field or if the field would extend beyond the right side of the display, the relocation will not take place. If you decide not to move the field, press EXIT to return to the previous menu.

MOVE OUTPUT — Allows you to move an output field in the same manner as moving an input field.

You can create, delete and move as many fields in the form as you wish. When you are satisfied with the number, length and location of the fields, press EXIT.

KEYBOARD — You should only use the softkey sets when you are creating and manipulating forms. If you use the keys on the keyboard, especially such keys as “Delete line”, “Insert line”, “Delete char”, and “Insert char”, you might get some unexpected results; your orders may be muddled and incoherent.

In-/Output Order

Assigning orders to the fields is necessary, since this information will be needed when the form is used by the application.

Input Order	The order in which the program will read data from the form.
Output Order	The order in which the program will write data to the form.
Tab Order	The order in which the cursor moves from one input field to the next when TAB is pressed.

Each input field must be assigned an input- and tab order, and each output field must be assigned an output order. If no order has been specified, a default order is assumed by the CFORM program.

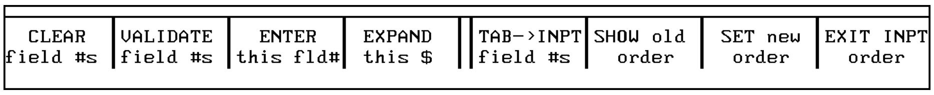
To specify a different input order, press the INPUT ORDER softkey. The form fields are redrawn with the input-field numbers in the input fields. The output fields are not shown.

To specify a different tab order, press the TAB ORDER softkey. The form fields are redrawn with the tab field-numbers in the input fields. The output fields are not shown.

To specify a different output order, press the OUTPUT ORDER softkey. The form fields are redrawn with the output-field numbers in the output fields. The input fields are not shown.

You can always see which order you are currently manipulating, since it is indicated in the EXIT softkey.

There is a softkey menu for each of the order (input, tab, and output). Here is an example of the input order:



```
  CLEAR  |VALIDATE | ENTER   | EXPAND  ||TAB->INPT|SHOW old |  SET new | EXIT INPT
field #s |field #s |this fld#| this $  ||field #s |  order  |  order   |  order
```

CLEAR field #s — Clears all the field numbers from the form.

VALIDATE field #s — Takes the numbers currently in the fields and verifies that they represent a correct sequence of order numbers. The sequence of numbers will be corrected if necessary. If bad numbers, such as “ABC” or “12.3.34”, are given, no such correction will take place. (Refer to page 23 for more details on numbering.) This softkey only validates numbers on the screen, it will not set the order numbers. If you want to store the form later with exactly these order numbers, use the SET new order softkey).

ENTER this fld# — Allows you to type in a number for a field, regardless of the size of the field. A $ sign is displayed in a field if the number requires more space than is available in this field. Position the cursor in the field to which you wish to assign a number. Press this softkey, and the prompt ENTER ORDER NO: ____________ …will be temporarily inserted in a line above the field. The relevant field will be marked “#”. Type in the new number and press RETURN. If the field is too small to display the entire number, a $ sign will be placed in that filed, showing that a number has been assigned to that field.

EXPAND this $ — This softkey allows you to view the $ numbers on the screen. A $ sign is visible in any field whose order number is too large to be displayed in that field. To view the number, position the cursor in the relevant field and press EXPAND this $. The number will be displayed in the lower part of the softkey on the screen.

TAB->INPT field #s — This softkey is only available when you are defining the input order. It will change the setting of the input order to be the same as the tab order, providing you with a frequently-used input convention in the form of a softkey.

INPT->TAB field #s — This softkey is only available when you are defining the tab order. It will change the setting of the tab order to be the same as the input order.

SHOW old order — This softkey displays the order which is currently SET. It may be different from the order displayed before you pressed this softkey. Use this softkey to view the original order after you have made some changes, but before you have made these changes permanent by pressing the SET new order softkey.

SET new order — When you press this softkey, the system first of all validates the numbers in the fields (see VALIDATE for more details). The resulting order after the system validation is complete now becomes the valid order. Note that when you STORE a form, the order is only changed if you press the SET new order softkey first. You can re-examine the current valid order by pressing the softkey SHOW old order.

EXIT order — This softkey returns you to the previous menu.

KEYBOARD — You use the keyboard to directly type in the relevant numbers in the fields, except for those fields which are too small to display that number. Use the ENTER this fld# softkey to input numbers which are too long for the respective field. You must not type the $ sign in a field from the keyboard.

Hints on Order Numbers

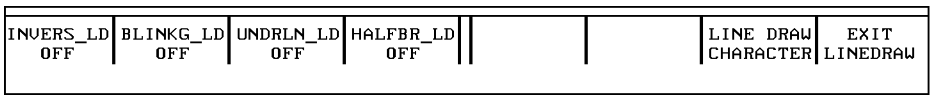
- A valid order-sequence for “n” fields starts at 1 and ends with “n”, consisting of positive integers.
- The default sequence of numbering is from top left to bottom right.
- Blank fields are given the value 0 prior to the automatic correction. If there are any gaps in the numbering sequence or if more than one field has the same number, corrections are made automatically.
- When the automatic corrections have been made, the process attempts to retain all unique numbers with values between 1 and “n”.
- Numbers with fractions can also be typed in. This means you can insert a new field where you want within the existing string.

Layout Functions

Use the LAYOUT FUNCTIONS softkey to draw the image of the form. The images are all non-field parts, such as descriptive information or line-draw images.

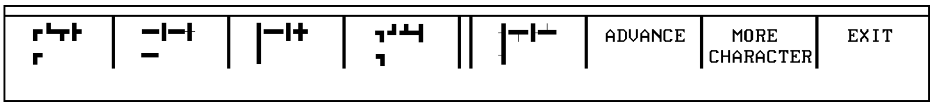
INVERSE HERE, BLINKING HERE, UNDERLINE HERE and HALF BRGHT HERE — Change the character accordingly at the current cursor position. For example, if you want to emphasize the form’s title by using inverse video, position the cursor at the first character of the title and press the softkey INVERSE HERE several times until all the characters are highlighted.

LINE DRAW ENHANCMT — When you press this softkey, a new set of softkeys appears, as shown here:



These keys allow you to specify the enhancements for line-draw characters only. The current status of the enhancement (on or off) is displayed in the lower part of the softkey on the screen. The EXIT softkey returns you to the layout softkeys.

LINE DRAW CHARACTRS — When you press this softkey, you gain access to the line-draw-character softkeys, as shown here:



The line-draw characters provided are common to the HP 700/92. The MORE CHARACTER softkey allows you to cycle through five sets of characters. The ADVANCE softkey selects one of the four line-draw characters within the line-draw softkeys. The selected character is displayed in the lower part of the line-draw-enhancement softkeys at the bottom of the screen. The EXIT softkey returns you to the layout softkey menu.

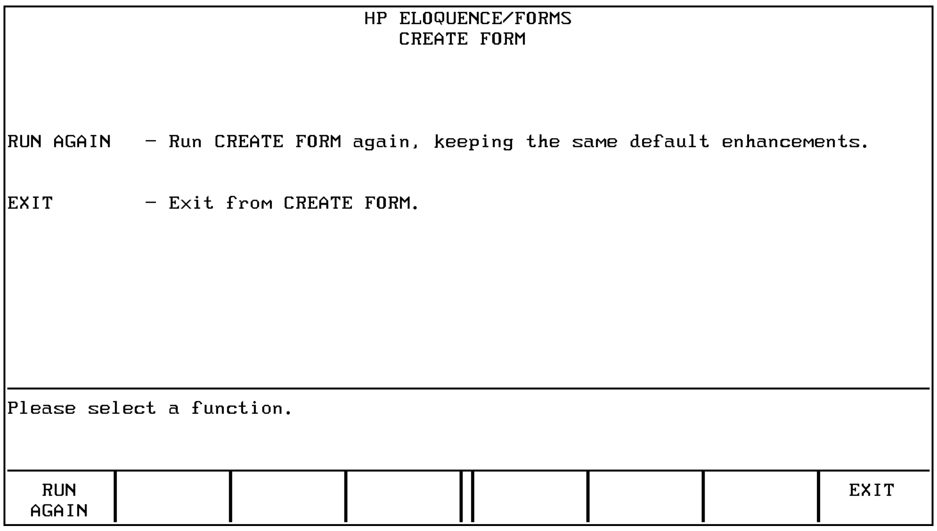
EXIT LAYOUT — Pressing this softkey returns you to the previous menu.

KEYBOARD — You use the keyboard to type in labels, descriptions, the name (title) of the form, in between the fields, but not in the fields themselves. Remember that you must not use the “Delete line”, “Insert line”, “Delete char”, or the “Insert char” keys to create or modify forms, since these keys might produce unexpected results; your orders may be muddled and incoherent.

Store the Form

When the STORE FORM softkey is pressed, you are asked for the name of the form. You specify the form file name which may include a unit specifier or volume label. If a file by that name already exists, you are asked if the old file is to be purged. If you answer no, you will be asked for a new file name. If you answer yes and the file is unprotected, a purge will be performed and the new form will be stored. If the file is protected, you cannot overwrite it.

After the form is stored, the keys become defined as follows:



```
                              HP ELOQUENCE/FORMS
                                 CREATE FORM


RUN AGAIN   - Run CREATE FORM again, keeping the same default enhancements.

EXIT        - Exit from CREATE FORM.

Please select a function.
```

If you wish to create another form, press RUN AGAIN. The keys become defined as they were when the program was first run. These keys are shown earlier in this section. The display enhancements are defined as you set them.

If you wish to exit the program, press EXIT.

Summary of CFORM

Now that you have read all the details on creating forms, here is a short overview. The order of operation shown here is not mandatory.

1 Run “CFORM”.

2 Using the Enhancements softkeys, set the default field video enhancements.

3 Press CREATE NEW FORM or CREATE FROM FORM.

4 Create the image of the form.

5 Create input and output fields.

6 Set input, tab and output order.

7 Store the form.

3

Modifying Forms

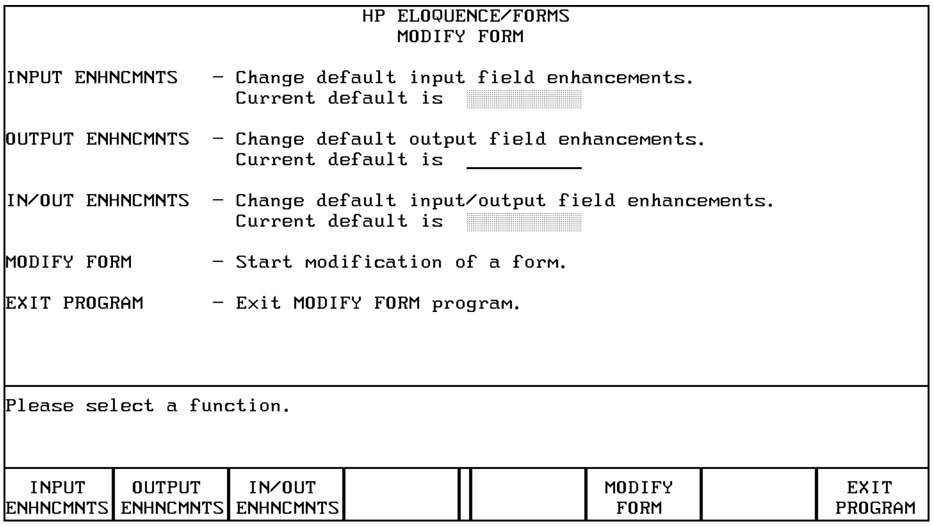
The Modify form (MFORM) program allows you to modify a form without destroying the program-link information (the number and length of the fields and the order in which they are accessed by the program). The program-link information cannot be changed with MFORM because any changes in the form would require similar changes in every program that uses the form.

To run the MFORM program, start Eloquence and type the following:

```
RUN"MFORM" RETURN
```

NOTE: For the MODIFY FORM menu to appear, a VOLUME statement pointing to the directory /opt/eloquence/share/prog must be included in the global configuration file eloq.config. The sample global configuration file d.eloq.config contains such a statement (**`VOLUME SYSTEM :Z2,7,0 /opt/eloquence/share/prog`**).

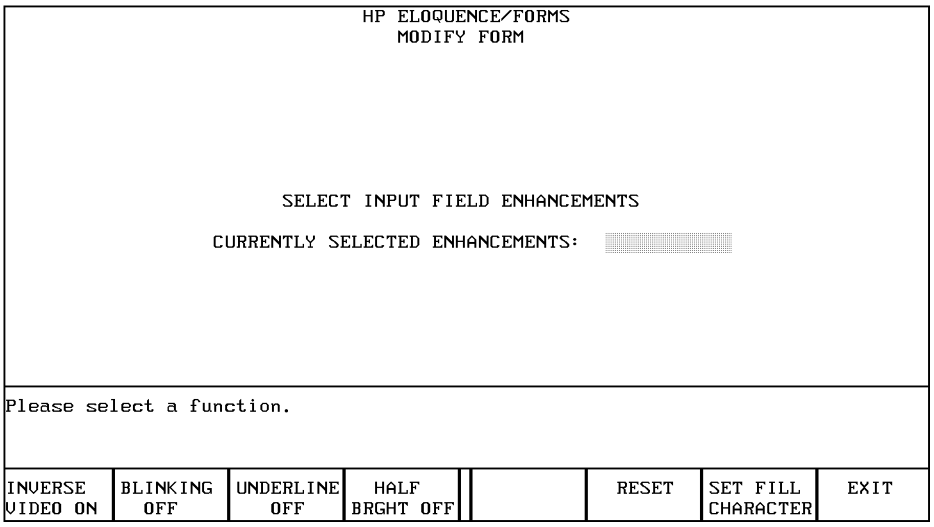
The current display enhancements (inverse video, half bright, underline) for input, output and input/output fields are shown along with the definitions of the softkeys:



```
                         HP ELOQUENCE/FORMS
                            MODIFY FORM

INPUT ENHNCMNTS   - Change default input field enhancements.
                    Current default is

OUTPUT ENHNCMNTS  - Change default output field enhancements.
                    Current default is

IN/OUT ENHNCMNTS  - Change default input/output field enhancements.
                    Current default is

MODIFY FORM       - Start modification of a form.

EXIT PROGRAM      - Exit MODIFY FORM program.

Please select a function.
```

The default field enhancements can be changed by pressing one of the softkeys marked ENHNCMNTS.

If the INPUT ENHNCMNTS softkey is pressed, the following menu is displayed:



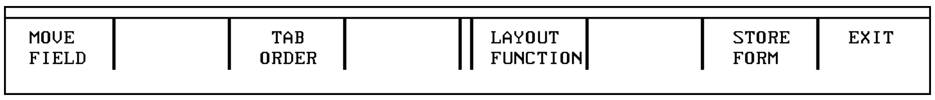
```
HP ELOQUENCE/FORMS
MODIFY FORM

SELECT INPUT FIELD ENHANCEMENTS

CURRENTLY SELECTED ENHANCEMENTS:

Please select a function.
```

Each softkey has the same definition and use as in the CFORM program:

SET FILL CHARACTER — Allows you to change the character which appears in the field while MFORM is running. This character becomes a blank when the form is stored.

INVERSE VIDEO, BLINKING, UNDERLINE and HALF BRGHT — Toggle on and off. They set the default display enhancement. Note, however, that each field does not have to use the default enhancement. The enhancements on an individual field can be changed when the rest of the form image is changed.

When MODIFY FORM is pressed on the initial MFORM menu, you are asked for the name of the form to be modified. The form will be displayed and the keys take on a new definition.



MOVE FIELD — Moves the input or output fields to new locations. However, an input/output field is moved as a whole. That is, the input or output portion of the field cannot be moved separately.

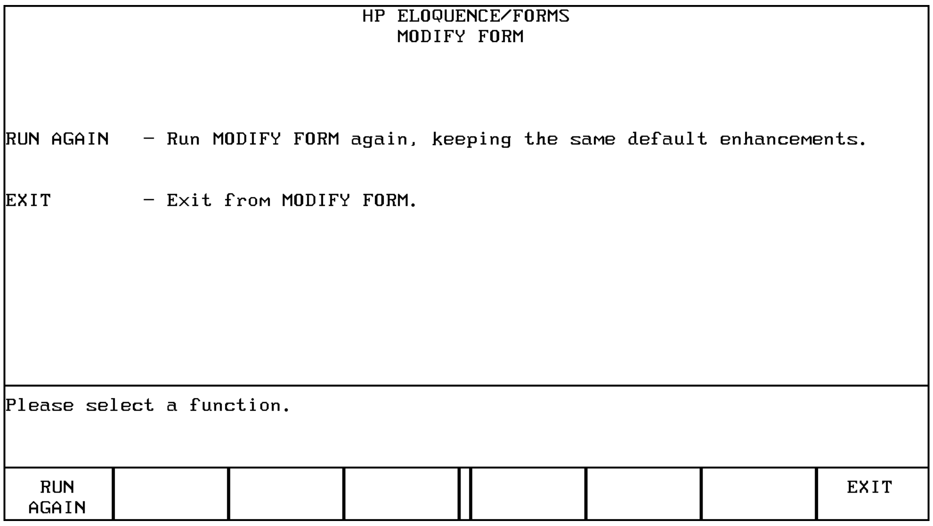
TAB ORDER — Sets the order the cursor moves. The order is set in the same manner as with CFORM.

Note that there is no key which allows you to create or delete fields or to change the order of input from or output to fields. If you want to change these properties, use the Create Form (CFORM) program.

LAYOUT FUNCTION — Used along with the keyboard to change the image of the form (refer to page 23).

STORE FORM — Stores the modified form. The old file is purged and the new form is stored with the same name as the old form. If the file is protected, you cannot overwrite it. If the file cannot be found, you are asked for a new file name.

Once the form is stored, the new menu is:



```
                              HP ELOQUENCE/FORMS
                                 MODIFY FORM


RUN AGAIN   - Run MODIFY FORM again, keeping the same default enhancements.

EXIT        - Exit from MODIFY FORM.
```

```
Please select a function.
```

To terminate the MFORM program, press the EXIT softkey.

To modify another form, press the RUN AGAIN softkey.

4

Interaction between Programs and Forms

Form Link Information

The linking information stored with the form consists of:

1 The number and location of input and output fields.

2 The order in which the input and output fields are accessed by the program.

The program that uses the form must be written so that the program-link is complete. The program must read or write the correct number of input or output fields in the correct order. If not, incorrect results will occur.

For example, assume the program uses the following form:

```
CUSTOMER NAME

PART NUMBER

PRICE            ___________

QUANTITY

SUB TOTAL        ___________

TAX              ___________

SHIPPING CHARGE  ___________

TOTAL            ___________
```

The user fills in the inverse video fields and the program fills in the underlined fields. Suppose the input order was 1) CUSTOMER NAME 2) PART NUMBER 3) QUANTITY. Further assume that the program was written to use the input fields in the following order: 1) CUSTOMER NAME 2) QUANTITY 3)PART NUMBER. You can see the problems that would result. Wrong parts would be shipped, and the total charges would be wrong.

When you are writing a program that uses a form, but do not know the input and output order of the fields, do the following:

1 Run CFORM.

2 Press CREATE FROM FORM.

3 Type in the name of the form the program will use.

4 Press INPUT ORDER, which will display the order of input.

5 Take note of the input order.

6. Press EXIT.
7. Press OUTPUT ORDER which will display the order of output.
8. Take note of the output order.
9. Press EXIT.
10. Terminate CFORM by pressing EXIT PROGRAM.

Do not alter the form and do not store it. While the form is displayed, you should also note the length of each field.

Input and Output Field Pointers

Two pointers, an **input field pointer** and an **output field pointer**, keep track of the current input and output fields. Each time an item is input to the program from an input field, the input field pointer is incremented. The same occurs for the output field pointer. The fields are accessed in the order specified when the form was created (refer to page 21).

Input and Output to the Form

The standard Eloquence input and output statements are described in the following sub-sections. The descriptions given apply when used on an active form. An active form is one that is displayed and has linking information in memory. Only one form can be active at a time. All these statements are fully described in the *Eloquence Manual*.

The INPUT Statement

INPUT [["*prompt*",] *input list*]

This syntax first outputs the prompt (either a ? or the prompt specified in the syntax) to the current output field and increments the output field pointer. The cursor then moves to the current **tab** field. When the user presses RETURN, the contents of the current field is input and the input field pointer is incremented. If more than one input item is in the item list, the next prompt (if any) is output. The output field pointer is incremented and the cursor moves to the next tab field. The input field pointer is incremented when an item is input to the program.

For example, assume the following form is used. The field following NAME is an input field. The field following SOCIAL SECURITY NUMBER is an output field.

```
NAME

SOCIAL SECURITY NUMBER  ____________
```

Before the form is used, the input field pointer (IF#) is 1 and the output field pointer (OF#) is 1. The statement INPUT Name$ causes the form to appear as follows:

```
NAME

SOCIAL SECURITY NUMBER ? ____________
```

That is, the prompt (?) is output to the output field number 1 and the cursor appears in input field number 1. IF #=1, OF#=2. The user types in a name and presses RETURN.

```
NAME  JOHN DOE

SOCIAL SECURITY NUMBER ? ____________
```

In this case, variable Name$ equals JOHN DOE, IF#=2 and OF#=2.

If the program attempted to output the social security number, an error would result because OF# is greater than the number of output fields.

If no parameters are specified in the INPUT statement, it will put the program in the input state and wait for RETURN to be pressed. No prompt is output. The input field pointer is not incremented. The INPUT statement should be followed by an ENTER statement in order for the program to receive the data input.

For example, assume a program uses the following form:

NAME

SOCIAL SECURITY NUMBER ____________

Before the form is used, IF#=1 and OF#=1.

When INPUT is executed, the cursor appears and the program waits for the user to enter a name and press RETURN.

NAME JOHN DOE

SOCIAL SECURITY NUMBER ____________

The IF#=1 and OF#=1, however, the program does not know the name that was entered.

The ENTER Statement

`ENTER` *`input list`*

ENTER inputs data from the display and continues program execution. The ENTER statement inputs the value of the current input field to the input list. The input field pointer is incremented. If there is more than one item in the input list, the next value, now pointed to by the input field pointer is the input and the input field pointer is again incremented.

For example, assume that this sequence is executed while the previous form is active:

```
300  INPUT
310  ENTER Name$
```

Before INPUT is executed, IF#=1 and OF#=1.

NAME JOHN DOE

SOCIAL SECURITY NUMBER ____________

After ENTER is executed, IF#=2, OF#=1 and Name$=JOHN DOE. But if the form appeared as follows immediately before the ENTER statement was executed (the input field is blank):

```
NAME

SOCIAL SECURITY NUMBER  ____________
```

Then after the ENTER statement is executed IF#=2, OF#=1 and Name$=" " (equal number of blanks as in the field).

The DISPLAY and PRINT Statements

```
DISP display list
PRINT print list
```

Each statement outputs to the current output field and increments the output pointer. If more than one item is in the list, the next item is output to the current output field and the output field pointer is incremented.

For example, assume that after the name is input, the program outputs the social security number. If an INPUT Name$ is used to input the name, the form looks as follows:

```
NAME JOHN DOE

SOCIAL SECURITY NUMBER ? ____________
```

Since IF#=2 and OF#=2, you cannot use the DISP statement here. But, if the INPUT with no prompt is used, the form looks as follows:

```
NAME JOHN DOE

SOCIAL SECURITY NUMBER  ____________
```

Since IF#=2 and OF#=1, the program can use the DISP statement.

```
160 DISP "111-11-1111"
```

```
NAME JOHN DOE

SOCIAL SECURITY NUMBER 111-11-111
```

Now IF#=2 and OF#=2.

The LINE DISPLAY Statement

```
LDISP display list
```

Moves the cursor to the first unprotected line following the form before outputting the display list. The output field pointer is not incremented.

Using the same example form shown previously, assume a program does not have a social security number for the name given, and wants the user to type in a number. The program can use LDISP.

```
300 LDISP "Please type in the social code."
```

`NAME` JOHN DOE

`SOCIAL SECURITY NUMBER` ____________

`Please type in the social security code. -`

After LDISP is executed, IF#=2 and OF#=1.

The LINE ENTER Statement

`LENTER` *`string variable`*

The LENTER statement immediately inputs the current line on the display and continues program execution. If the cursor is in the form, an error occurs. Execution of LENTER does not affect the field pointers.

The LINE INPUT Statement

$$\text{LINPUT} \left[\text{”}\textit{prompt}\text{”} \begin{Bmatrix} ; \\ , \end{Bmatrix} \right] \textit{string variable}$$

When executed, LINPUT moves the cursor to the first unprotected line following the form and outputs a line-output prompt. The entire line is returned when RETURN is pressed. The input field pointer is not incremented.

In the two previous examples, you saw how one statement, LDISP, would output without affecting the form or form pointers and how another statement, LENTER, would input. Instead of using these two statements, the program could use LINPUT.

Assume the form and display looks as follows:

`NAME` JOHN DOE

`SOCIAL SECURITY NUMBER` ____________

Now, IF#=2 and OF#=1.

The program executes the following statement.

```
100  LINPUT ”please type in the social security code.”,Ssnum$
```

The form and display become:

NAME JOHN DOE

SOCIAL SECURITY NUMBER ____________

Please type in the social security code. -

Because a comma followed the prompt in the statement, the cursor moves to the next line. The user types in a number and presses RETURN. The IF#=2 and OF#=1.

If a semicolon (;) follows the LINPUT prompt, the cursor remains on the same line as the prompt. Then when RETURN is pressed, the entire line, including the prompt, is assigned to Ssnum$.

TAB and RETURN

When the program is waiting for input, either TAB or RETURN can be used. The TAB key moves the cursor from one input field to the next. It does not take the program out of the waiting state. The RETURN key does not tab through the fields. It returns the program to the execution state. If the next statement is an input statement, the cursor is displayed in the next tab field and it appears that RETURN has tabbed to the next input field.

5

Controlling Forms

Seven statements and a function are available to control use of forms:

GET FORM	Display a new form on the screen.
CLEAR FORM	Erases the input and output fields and resets field pointers.
CURSOR	Sets a value for the input and/or output field pointers.
TFNUM	Returns the tab position of the cursor.

EXIT FORM	Breaks the link between the form and the program.
DELETE FORM	Erases the form from the display and breaks the program-form link.
LOAD FORM	Displays a form on the screen and loads information into memory about the fields of the form (for example, length, type, and location).
STORE FORM	Used to store a LOADed form.

NOTE: The form control statements (except for LOAD FORM and STORE FORM) are available in the development *and* run-time versions of Eloquence. LOAD FORM and STORE FORM are *only* available in the development version of Eloquence.

The GET FORM Statement

The GET FORM statement displays a new form and loads the program-form linking information into memory. The syntax is:

GET FORM "*form name* [*volume*]"

Form name is the name of the form. Volume spec is a volume label or unit specifier.

When a form is displayed and the linking information is loaded, the form is active. To ensure that only one form is active at a time, GET FORM performs an EXIT FORM before the new form is activated.

The program uses GET FORM initially to display a form to be used for input and output or to change from one form to another. Once a form is activated, all item input and output is done through that form. (Item input and output is done with the INPUT, ENTER, DISP and PRINT statements.) Line output and input (LDISP, LENTER and PRINT) is done outside the form image and does not affect the form pointers.

The CLEAR FORM Statement

The CLEAR FORM statement clears the input and output fields of the current form. Use CLEAR FORM to reuse a form for further input and output. The syntax is:

```
CLEAR FORM
```

When CLEAR FORM is used, the form on the display remains as it is; only the input and output fields are erased.

Execution of this statement resets the field pointers to the first input field and first output field of the form. The cursor is placed in the upper-left corner of the form. The form remains active and the link between program and form is not altered.

The CURSOR Statement

Use the CURSOR statement whenever an input or output operation is wanted in a particular field that is out of the order specified when the form was created.

The CURSOR statement can be used to set values for the input and output field pointers via the IF# (Input Field number) and OF# (Output Field number) parameters. The cursor can be set to a particular input field with the CF# parameter. Cursor parameters are introduced in the Output Operations chapter in the *Eloquence Manual*. However, when a form is active, some additional parameters can be used. The general syntax of these parameters is:

$$\texttt{CURSOR} \quad \left\{ \begin{array}{l} ,\text{IF \# numeric expression} \\ ,\text{CF\# numeric expression} \\ ,\text{OF\# numeric expression} \end{array} \right\} \texttt{, . . .}$$

The IF# parameter sets the input field pointer to the value specified in the numeric expression following IF#. When the next input operation is executed, the cursor is moved to the first character of that field. OF# sets the output field pointer to the value specified in the numeric expression following OF#. When the next output operation is executed, the cursor moves to that output field.

For example, to set the input field pointer to input field number 20 and the output field pointer to output field number 5:

```
CURSOR IF#20, OF#5
```

The CF# parameter is normally used with the IF# parameter. Using CF# causes the cursor to move to the specified input field rather than to the field specified in the IF# expression.

For example, assume the program determines that a given input is incorrect and that the user must re-enter the information in that field. The IF# parameter could be used to set the input field pointer and the cursor to the incorrect field. However, the user may use TAB to move the cursor and correct another field. The program would not receive this change. A better way is for the program to set the IF# equal to one and CF# to the incorrect field. Then the cursor would move to the incorrect field and the input field number is one. After the user enters any information, the program rereads all the input fields.

TFNUM Function

The function TFNUM returns the tab field number of the current tab position. It returns zero if the form is not active, or before the first position. A program can use this to test if the user has filled in all the fields of the form. For example:

```
10    INPUT
20    IF TFNUM%<4 THEN 10
30    ENTER A(1), A(2), A(3), A(4)
```

The user presses TAB to tab to each field. Pressing TAB in the last field will cause line 30 to be executed.

The EXIT FORM Statement

The EXIT FORM statement breaks the link between the active form and the program. The syntax is:

```
EXIT FORM
```

The cursor is moved to the first character of the first line after the form. The form is no longer active.

After EXIT FORM is executed, the form is still displayed, but further input and output operations do not use it. The lines of the form are unprotected.

The DELETE FORM Statement

The DELETE FORM statement breaks the link between the active form and the program. In addition, it erases the form from the display. The syntax is:

```
DELETE FORM
```

The LOAD FORM Statement

The LOAD FORM statement is only available in the development version of Eloquence. This statement displays a form on the screen and loads information into memory about the fields of the form (for example, length, type, and location). Syntax is as follows:

LOAD FORM "*form name* [*volume spec*]", *array*

Replace *form name* with the name of the form. The *volume spec* parameter is optional. If used, replace it with the volume label or unit specifier. Replace *array* with an integer array name. This array contains information about the fields of the form. The minimum array size (number of elements in the array) is equal to 7 + 2 $\ast$ number of input fields + number of output fields. So, if you have 50 input fields and 70 output fields, the array associated with the form must allow for at least 177 elements.

LOAD FORM is useful for automatic forms generation or modification. For example, suppose you have a database that contains a record named CUSTOMER NUMBER, and it has a field length of 16. Suppose you change the database schema text file so this field is 20 characters long. The screens that contain the CUSTOMER NUMBER record must be changed. This can be done using a program containing the LOAD FORM and STORE FORM statements. The general flow of the program would be (1) LOAD FORM, (2) search for CUSTOMER NUMBER, (3) change field length to 20, (4) STORE FORM.

Error Messages

Listed below are the error numbers that could occur when the LOAD FORM statement is executed. Next to each number is an explanation.

`ERROR 56` - File or directory not found or no read permission.

`ERROR 861` - Improper number of elements. The array does not have enough elements to hold the data from the LOAD FORM statement.

`ERROR 862` - Improper array type. The array specified must be an integer array.

The STORE FORM Statement

The STORE FORM statement is only available in the development version of Eloquence. This statement stores a form that has been loaded using the LOAD FORM statement. Syntax is as follows:

STORE FORM "*form name* [*volume spec*]", *array*

Replace *form name* with the name of the form. The form name can be up to nine characters long; the extension .FORM is automatically appended to the form name. The *volume spec* parameter is optional. If used, replace it with the volume label or unit specifier. Replace *array* with an integer array name. This array will contain the information about the fields of the form. The minimum array size (number of elements in the array) is equal to 7 + 2 $\ast$ number of input fields + number of output fields. So, if you have 50 input fields and 70 output fields, the array associated with the form must allow for at least 177 elements.

A form can be stored back to its LOADed name or to a new form name using the STORE FORM command.

It is possible to store default values in the fields of a form. To do so, load the form (LOAD FORM), type information into the desired fields, and store the form (STORE FORM). Now when GET FORM is executed the information previously typed in and stored appears in the form on the screen.

Error Messages

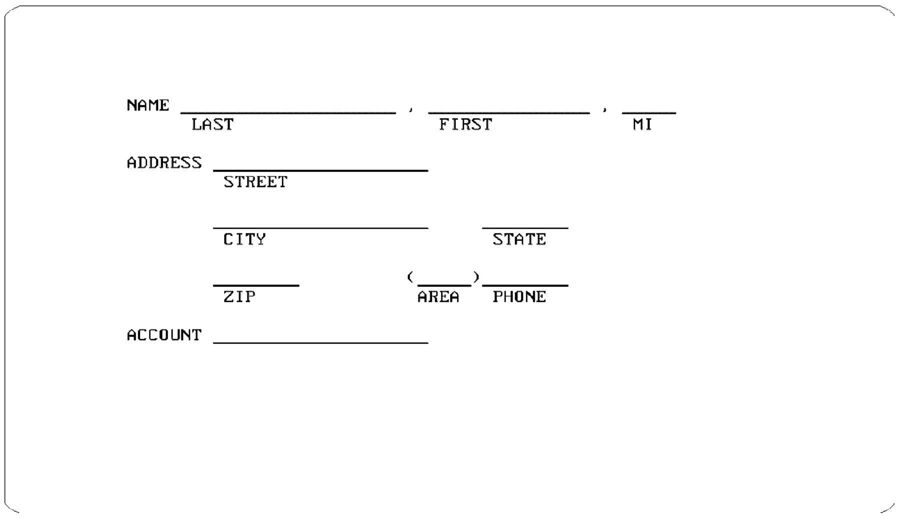
The error messages mentioned under page 50 also apply to STORE FORM. However, there is one additional message unique to STORE FORM. It is as follows:

`ERROR 863` - Inconsistant information given.

This message occurs if you update the forms image without updating the array elements or vice versa.

Example Program

When this program is run, it displays a form and asks the user to fill it in. When ENTER is pressed, the program checks that the zip code, area code and phone fields have been filled in with numbers. If not, the program resets the cursor and asks that the field be re-entered. The program uses a form that looks as follows:



```
NAME    ____________________ , _______________ , _____
            LAST                   FIRST             MI

ADDRESS ____________________
          STREET

        ____________________      ________
          CITY                      STATE

        ________          (_____)________
          ZIP              AREA   PHONE

ACCOUNT ____________________
```

```
10   !    The form is stored in file “NEWCUS”.
20   !    The input fields are name, address and phone.
30   !    The output field is account.
40   !   The customer information is stored in a file called “CUSTMR”
50   !   The first record contains the next account number to be used.
60   !
70   !    Begin program
80   !
90        OPTION BASE 1
100       DIM A$(10)[20],B$[80],C$[200]
110       ASSIGN #1 TO “CUSTMR,FILES”
120  !
130       GET FORM “NEWCUS,FILES”
140  !
150 Input: INPUT
160       IF TFNUM%<>9 THEN GOTO 150
170           ENTER A$(*)
180  !
190 A:    IF (A$(7)>”99999”) OR (A$(7)%<”00000”) THEN GOTO Zip
200 B:    IF (A$(8)>”999”) OR (A$(8)%<”000”) THEN GOTO Area
210 C:    IF (A$(9)>”9999999”) OR (A$(9)%<”0000000”) THEN GOTO Phone
220  !
230       READ #1,1;Acc
240       PRINT Acc
250       B=Acc+1
```

```
260      PRINT #1,1;B
270      A$(10)=VAL$(Acc)
280 !
290      FOR I=1 TO 10
300        C$=C$+A$(I)
310      NEXT I
320 !
330      PRINT #1,B;C$
340      LDISP “DO YOU WANT TO ADD ANOTHER CUSTOMER?”
350      LINPUT B$
360 !
370      IF (B$[1]%<”Y”) OR (B$[1]>”Y”) THEN GOTO End
380 !
390      C$=” “
400      CLEAR FORM
410      GOTO Input
420 !
430 Zip: LDISP “YOU HAVE ENTERED AN INCORRECT ZIP CODE. PLEASE RE-
ENTER.”
440      CURSOR IF#1,CF#7
450      GOTO Input
480 !
490 Area: LDISP “YOU HAVE ENTERED AN INCORRECT AREA CODE. PLEASE RE-
ENTER.”
500      CURSOR IF#1,CF#8
510      GOTO Input
540 !
550 Phone: LDISP “YOU HAVE ENTERED AN INCORRECT PHONE NUMBER.”
560      LDISP “PLEASE RE-ENTER”
570      CURSOR IF#1,CF#9
580      GOTO Input
610 !
620 !    End of program
630 !
640 End: EXIT FORM
650      ASSIGN #1 TO *
660      END
```

6

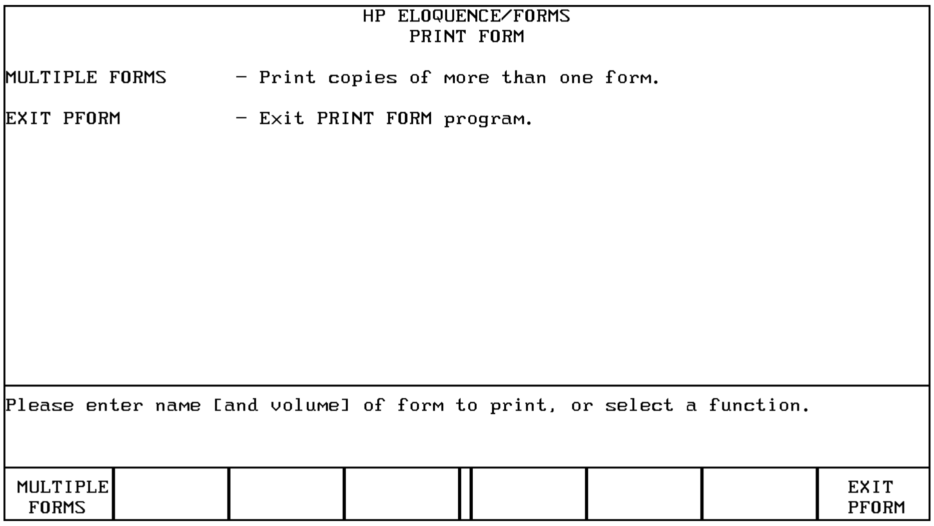
Printing Forms

To print a form, use the PFORM (print form) program. PFORM is supplied with the development version of Eloquence. To run PFORM, start Eloquence and type the following:

```
RUN ″PFORM″ RETURN
```

NOTE: For the PRINT FORM menu to appear, a VOLUME statement pointing to the directory /opt/eloquence/share/prog must be included in the global configuration file eloq.config. The sample global configuration file d.eloq.config contains such a statement (**`VOLUME SYSTEM :Z2,7,0 /opt/eloquence/share/prog`**).

The initial PFORM menu asks for the name of the form to be printed:



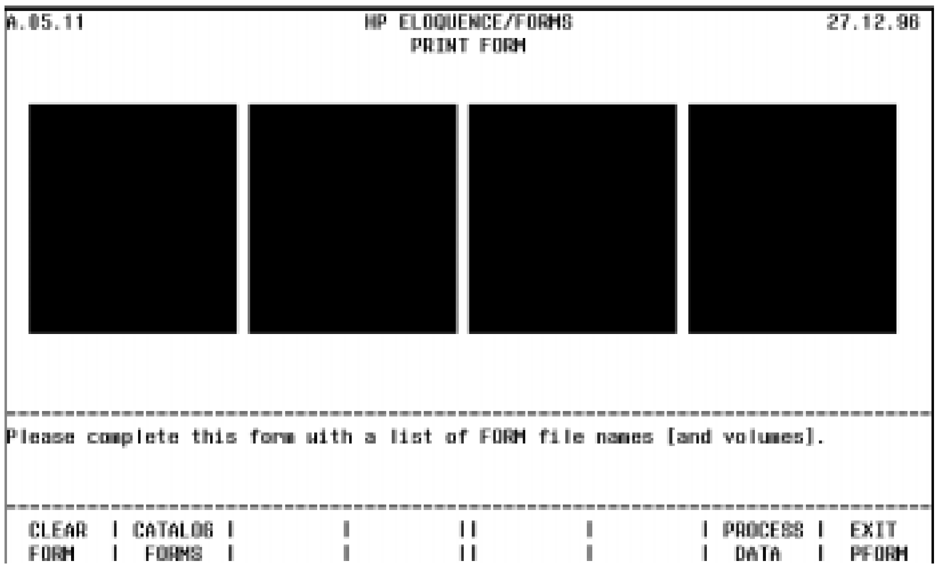
```
                                HP ELOQUENCE/FORMS
                                    PRINT FORM

MULTIPLE FORMS      - Print copies of more than one form.

EXIT PFORM          - Exit PRINT FORM program.
```

```
Please enter name [and volume] of form to print, or select a function.
```

To print copies of one form, enter its file name and optionally its volume. Then skip the next section and read page 60 . If copies of more than one form are to be made, press the MULTIPLE FORMS softkey.

Multiple Forms

When MULTIPLE FORMS is pressed, the following menu is displayed to allow you to enter one or more form file names:



```
A.05.11                          HP ELOQUENCE/FORMS                          27.12.98
                                     PRINT FORM

--------------------------------------------------------------------------------------
Please complete this form with a list of FORM file names [and volumes].

--------------------------------------------------------------------------------------
  CLEAR   | CATALOG |         |         ||         |         | PROCESS |  EXIT
  FORM    |  FORMS  |         |         ||         |         |  DATA   |  PFORM
```

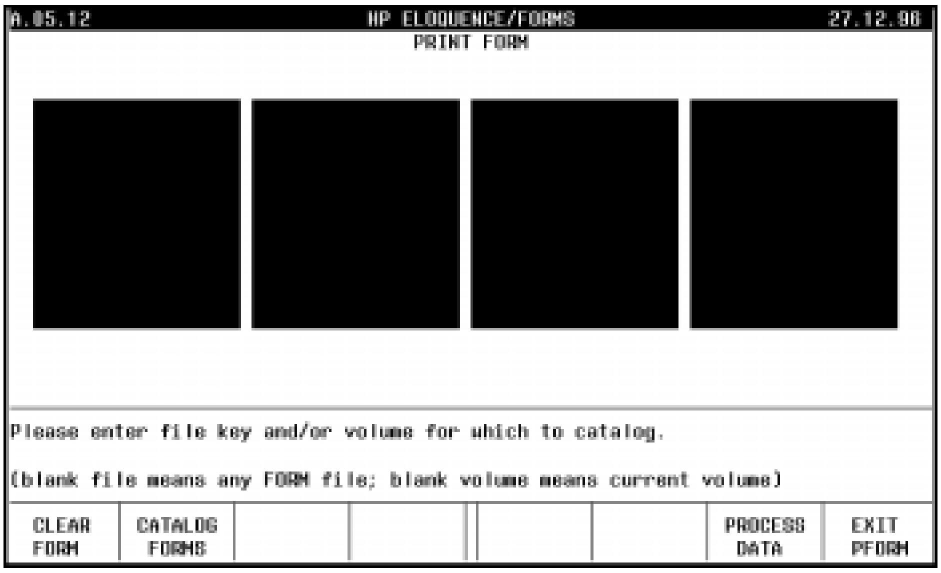
You may enter each file name and volume or press CATALOG FORMS to display selected file names automatically.

CLEAR FORM — clears the form for new entries.

CATALOG FORMS — asks for a file key (first letter(s) of the file name) and/or a volume.

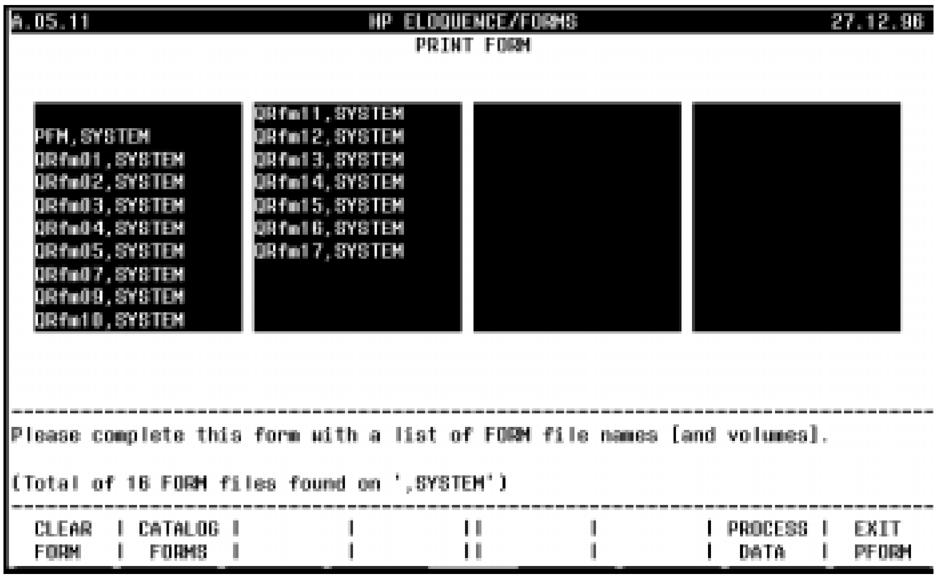
When the CATALOG FORMS softkey is pressed, the following screen is displayed:

If only a volume is specified, either a comma precedes the volume name or a colon precedes a unit specifier. A blank volume indicates the current MSI.



```
A.05.12                    HP ELOQUENCE/FORMS                    27.12.98
                               PRINT FORM

Please enter file key and/or volume for which to catalog.

(blank file means any FORM file; blank volume means current volume)
```

All form files with the correct file key and on the specified volume are displayed, as shown by the following screen:



```
A.05.11                    HP ELOQUENCE/FORMS                    27.12.98
                               PRINT FORM

                      QRfm11,SYSTEM
 PFM,SYSTEM           QRfm12,SYSTEM
 QRfm01,SYSTEM        QRfm13,SYSTEM
 QRfm02,SYSTEM        QRfm14,SYSTEM
 QRfm03,SYSTEM        QRfm15,SYSTEM
 QRfm04,SYSTEM        QRfm16,SYSTEM
 QRfm05,SYSTEM        QRfm17,SYSTEM
 QRfm07,SYSTEM
 QRfm09,SYSTEM
 QRfm10,SYSTEM

Please complete this form with a list of FORM file names [and volumes].

(Total of 16 FORM files found on ',SYSTEM')
```

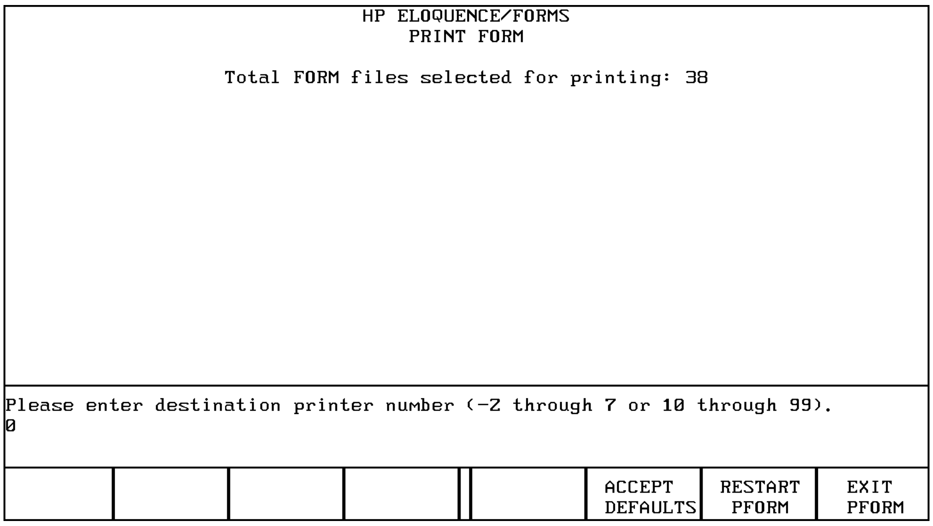
You may use the edit keys (CLEAR, INSERT, etc.) to add or delete files from the list.

PROCESS DATA — when the list is complete press this softkey. A new menu is displayed for printer selection.

EXIT PFORM — exits the program without printing any forms.

Printer Selection

After one or more form names are entered, PFORM allows you to choose a printer.



```
                          HP ELOQUENCE/FORMS
                              PRINT FORM

                 Total FORM files selected for printing: 38

Please enter destination printer number (-2 through 7 or 10 through 99).
0
```

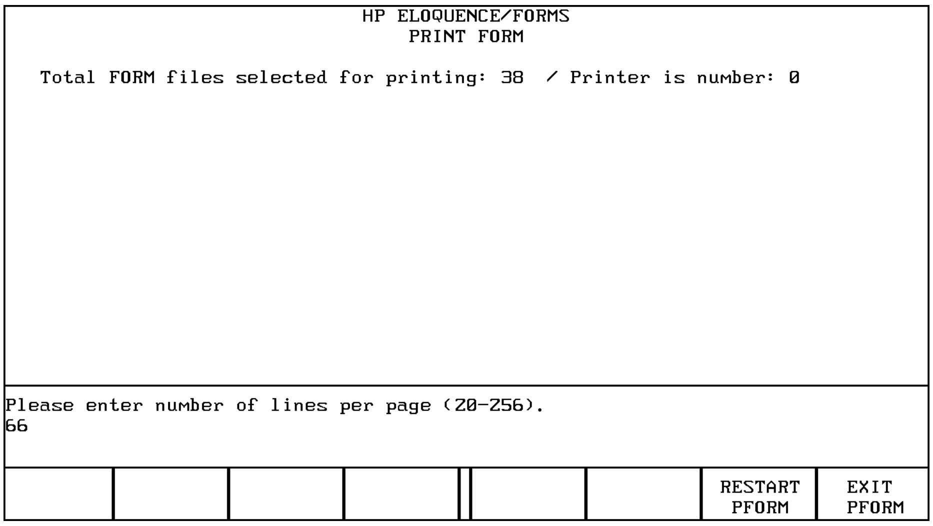
Enter the printer number. Note that forms cannot be printed on the terminal. (To view the form on the terminal use MFORM.)

ACCEPT DEFAULTS — immediately prints one copy of the form using default fill characters but without headers. The Without Order version is printed (refer to page 64). If you run PFORM and change the defaults then return to this menu, the current defaults are those you set. Exiting the program resets the defaults.

RESTART PFORM — returns to the multiple forms menu.

EXIT PFORM — exits the program without printing any forms.

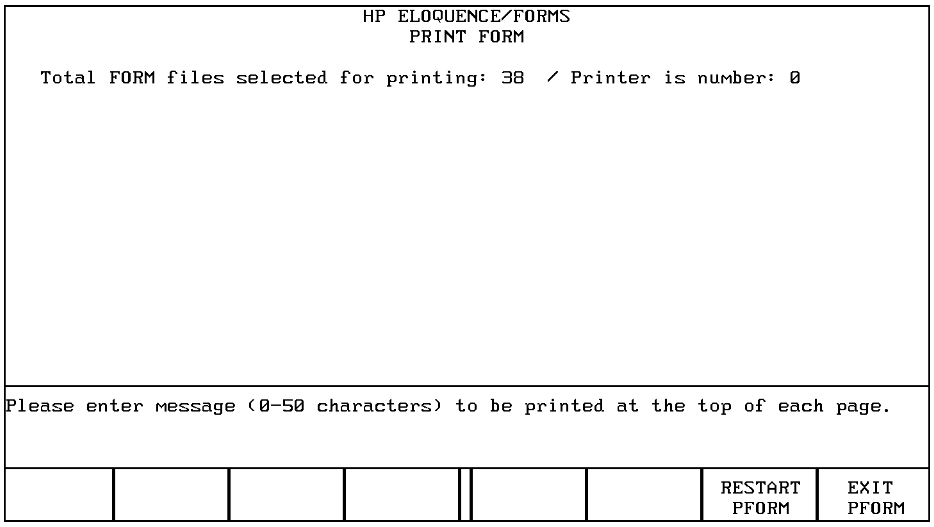
When you have selected the printer, the following screen appears to request the desired number of lines per page for the printout:



```
                    HP ELOQUENCE/FORMS
                        PRINT FORM

   Total FORM files selected for printing: 38  / Printer is number: 0

Please enter number of lines per page (20-256).
66
```

The default number of lines per page is 66. If you do not enter another value, this default value will be used by PFORM.

When you have selected the number of lines per page, you may also enter a message to be printed along with the form at the top of every page. The following screen will appear to request the entering of the optional message:



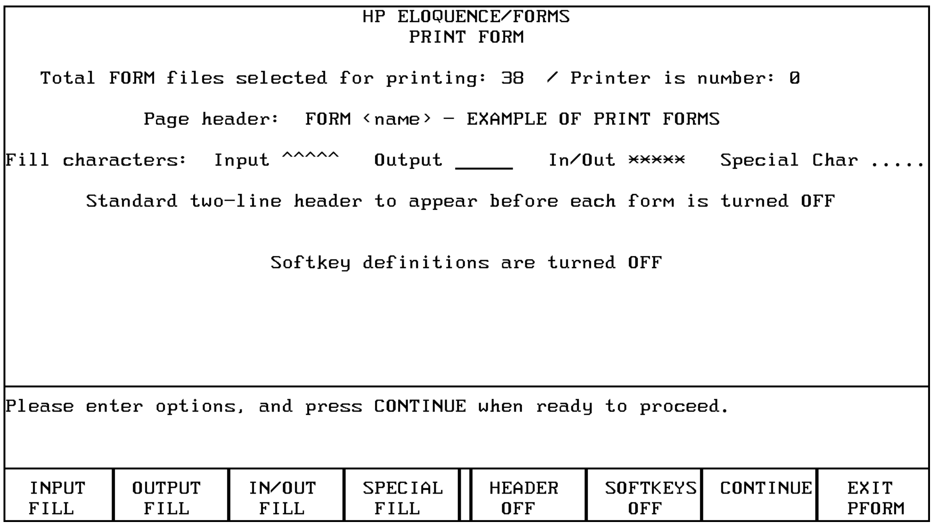
```
                    HP ELOQUENCE/FORMS
                        PRINT FORM

   Total FORM files selected for printing: 38  / Printer is number: 0

Please enter message (0-50 characters) to be printed at the top of each page.
```

Display Enhancements and Fill Characters

When the form is displayed on the terminal, special characters such as the line drawing set are used to enhance the form. Most printers cannot print these characters, so an alternate character may be selected. If a character other than blank is selected, that character replaces all special characters. (A special character is any character which is not on the local language keyboard.) If a blank is specified for special characters, the printer will attempt to print all special characters.

You may also select a character to be used as a fill character for the different types of fields. If a blank is entered, blanks are printed.

After the printer and message are selected, the following menu allows these character selections:



```
                                HP ELOQUENCE/FORMS
                                    PRINT FORM

   Total FORM files selected for printing: 38  / Printer is number: 0

            Page header:  FORM <name> - EXAMPLE OF PRINT FORMS

Fill characters:  Input ^^^^^   Output _____   In/Out *****   Special Char .....

       Standard two-line header to appear before each form is turned OFF


                       Softkey definitions are turned OFF




Please enter options, and press CONTINUE when ready to proceed.
```

The default values for all fill characters are shown on the menu.

INPUT FILL — changes the default printer input field fill character.

OUTPUT FILL — changes the default output field fill character.

IN/OUT FILL — changes the default input/output field fill character.

SPECIAL FILL — changes the default character which appears anywhere on the form (except in fields) where a non-printable character appears.

HEADER OFF — enters or modifies a two-line header to be printed above each form. The softkey definition changes to HEADER ON. If pressed a second time, it returns to its initial definition.

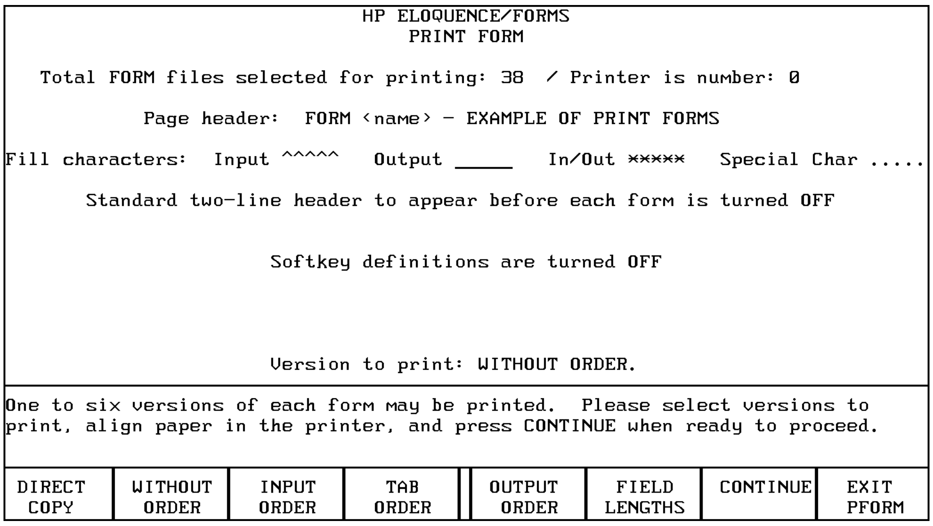
SOFTKEYS OFF — enters softkey definitions to be printed below each form.

CONTINUE — selects the version of the forms to be printed.

EXIT PFORM — exits the program without printing any forms.

Version Selection

After the fill options are selected, pressing CONTINUE displays the following screen:



```
                              HP ELOQUENCE/FORMS
                                  PRINT FORM

   Total FORM files selected for printing: 38  / Printer is number: 0

            Page header:  FORM <name> - EXAMPLE OF PRINT FORMS

Fill characters:  Input ^^^^^   Output _____    In/Out *****   Special Char .....

       Standard two-line header to appear before each form is turned OFF


                       Softkey definitions are turned OFF




                       Version to print: WITHOUT ORDER.
-------------------------------------------------------------------------------
One to six versions of each form may be printed.  Please select versions to
print, align paper in the printer, and press CONTINUE when ready to proceed.

-------------------------------------------------------------------------------
 DIRECT | WITHOUT | INPUT | TAB  || OUTPUT | FIELD   |CONTINUE| EXIT
 COPY   | ORDER   | ORDER | ORDER|| ORDER  | LENGTHS |        | PFORM
```

Select the version of the form to be printed.

DIRECT COPY — prints a copy as it appears on the terminal. Any enhancements which cannot be printed are left out. No additional enhancements or fill characters are added.

WITHOUT ORDER — prints a copy using any fill characters or special character specified. This is the default copy which is printed if ACCEPT DEFAULTS is pressed.

INPUT ORDER — the fill character and the special character specified are used. A number is placed in each input field to indicate the input order. If the number has more digits than the field length, no number is placed in the field.

TAB ORDER — the fill characters and the special characters specified are used. A number is placed in each input field to indicate the tab order. If the number has more digits than the field length, no number is placed in the field.

OUTPUT ORDER — the fill characters and the special character specified are used. A number is placed in each output field to indicate the output order. If the number has more digits than the field length, no number is placed in the field.

FIELD LENGTH — the fill characters and the special character specified are used. A number is placed in each field to indicate its length.

CONTINUE — causes printing to begin.

EXIT PFORM — exits the program without printing any forms.

Printing Forms

Printing begins by pressing CONTINUE on the version selection menu. The forms are printed in the order specified. All selected versions of one form are printed before the next form is printed. While the forms are being printed the softkeys are defined as follows:

RESTART PFORM — returns to the multiple forms menu. All forms which have not been printed are listed.

EXIT PFORM — terminates PFORM.

After the forms are printed, the initial menu is displayed. You may enter another form name or exit PFORM by pressing EXIT PFORM.

Errors If an error (such as the form is not found) occurs during printing, the form name is displayed along with the error message. You may enter a correction (such as a different volume name) or press the softkey labeled SKIP FORM. Error messages are described in page 67 .

A

Error Messages

CFORM and MFORM Error Messages

The error messages that may occur when using the CFORM (create form) or MFORM (modify form) programs are listed below.

VOLUME (DIRECTORY) IS WRITE-PROTECTED — You have tried to write to a volume that is protected. Specify another volume for the write operation, or change the access rights for the directory associated with the volume to allow write access.

FATAL ERROR xxx ENCOUNTERED xxxx — An internal error occurred in a forms program. Note this error and report it to your service representative.

FIELD BEYOND COLUMN 255 — A field cannot begin after column 255.

FIELD DELETED ILLEGALLY — This error occurs during MFORM if you delete a field with the editing keys. This error causes the program to terminate. There is no recovery.

FIELD MOVED OR DELETED — This error occurs during CFORM if the system cannot find the field you are trying to move during the move field operation.

FIELD WILL NOT FIT ON LINE — You are trying to move a field to a position where the end of the field will be off the screen. Pick a new location and move the field there.

FILE “filename” **ALREADY EXISTS** — You have tried to store a form into a file that already exists. You can purge the old file and store the form, or store the form in another file or with the same name on a different volume. This error only occurs with the CFORM program.

FORMS LIMITED TO 255 LINES — The maximum size of a form image is 255 lines. If a larger form is necessary, use two forms.

ILLEGAL FIELD OVERLAP — You are trying to move a field to a position where it would overlap another field. You must pick a new location to which to move the field.

ILLEGAL POSITION FOR IN/OUT FIELD — You are trying to move an input/output field to a location where there would be no non-field character before and after the input/output field. Pick a new location for the field.

IMPROPER FILE NAME — The name you specified is an illegal file name.

IMPROPER FILL CHARACTER ENTERED — The system reserves some characters for its use (for example, $\leftarrow$, $\uparrow$, etc.). You cannot use these for fill characters.

IMPROPER PROGRAM ENTRY — You tried to enter a forms program at its entry point; CFRM, MFRM; and the program could not be found.

IMPROPER VOLUME LABEL OR UNIT SPECIFIER — You specified a volume label or unit specifier with illegal characters. A volume name is a maximum of eight characters. A unit specifier has a syntax of:

:*letter select code, device address, unit number*

This syntax is completely described in the *Eloquence Manual*.

INVALID FIELD NUMBER — This error may occur during an ordering operation when a field contains an invalid number (e.g., 1.1.1). The cursor will go to the first offending field. Enter a valid number. Check that all other fields contain valid numbers and try the operation again.

NO FILE “filename” **EXISTS** — The system cannot find the file. If you used a unit specifier, try again with the volume label. Otherwise, exit the program and execute CAT, which lists the files currently stored.

OVERLAY FILE REVISION LEVEL CONFLICT — An earlier version of CFRM or MFRM was loaded during an overlay operation.

POSITION WOULD JOIN TWO FIELDS — You are trying to move a field to a position where there would be no non-input or output field character between it and a like field. Two input fields must have at least one non-input field character between them, similarly for output fields.

PROGRAM FILE REVISION LEVEL CONFLICT — An earlier version of a forms program was loaded while the program was running.

SPECIFIED VOLUME NOT FOUND — The volume specifier you gave does not match any of the available volume specifiers. Check that you entered the volume specifier correctly.

TOO MANY INPUT FIELDS — You have tried to create more than 200 input fields. If more fields are needed, use two forms.

TOO MANY OUTPUT FIELDS — You have tried to create more than 200 output fields. If more fields are needed, use two forms.

PFORM Error Messages

These error messages may occur when using PFORM:

A NUMBER –2 through 7 or 10 through 99 IS REQUIRED — a device address outside the acceptable range was entered.

BLANK INPUT NOT ALLOWED — a file name or device address must be entered.

CANNOT CONTINUE UNTIL AT LEAST ONE VERSION IS SELECTED — select at least one version of the form to be printed by pressing the appropriate softkey. Then press CONTINUE.

ENHANCEMENTS/SPECIAL CHARACTERS NOT ALLOWED — video enhancements and special characters cannot be output by most printers. Re-enter using standard characters.

FATAL ERROR xxx ENCOUNTERED xxx — an internal error occurred in the PFORM program. Note this error and report it to your service representative.

FILE NAME IS REQUIRED — a file name must accompany the volume specifier. To enter all forms on a volume, use the multiple forms menu.

FILE NAME TOO LONG — the file name must be nine characters or less.

FILE NOT FOUND — the file was not found on the volume specified. If no volume was specified, the current mass storage volume was used.

GIVEN (OR DEFAULT) VOLUME NOT MOUNTED — a directory with the specified label was not found.

ILLEGAL PROGRAM ENTRY — attempt to enter the PFORM program at its entry point, PFRM, and the program could not be found.

IMPROPER VOLUME NAME — a volume name has a maximum of eight characters.

PFORM/PFRM REVISION NUMBER CONFLICT — PFORM and PFRM do not have the same revision number.

PRINTER IS MISSING, OR WRONG TYPE, DOWN, OR OFFLINE — check the status of the printer. RESET and switch the printer ON-LINE. Then re-enter the printer device address.

VOLUME NOT FOUND — the volume name specified is not found.

Program Error Messages

Errors which may occur during execution of a program that uses a form are listed below.

290	NOT ALLOWED WHEN FORM IS ACTIVE — The operation the program is trying to perform could destroy the integrity of the form (i.e., protect lines, unprotected lines, etc.).
291	NOT ALLOWED WITHIN FORM IMAGE — The program is trying to modify the form (i.e., creating unprotected lines within the form).
292	ATTEMPT TO INPUT AFTER LAST FIELD OF FORM — The input field pointer has a value greater than the number of the input fields. The program can clear the form, use the CURSOR statement to reset the value of the input field pointer, or a line input statement could be used.
293	ATTEMPT TO OUTPUT AFTER LAST FIELD OF FORM — The output field pointer has a value greater than the number of output fields. The program can clear the form, reset the output field pointer with the CURSOR statement, or use a line output statement.
294	NOT ALLOWED UNLESS FORM IS ACTIVE — The program is trying to execute a statement that operates on a form (i.e., CURSOR IF#, CF#, or OF#) and no form is currently active.
861	IMPROPER NUMBER OF ELEMENTS — The array does not have enough elements to hold the data from the LOAD FORM statement.
862	IMPROPER ARRAY TYPE — The array specified in a LOAD FORM or STORE FORM statement must be an *integer* array.
863	INCONSISTENT INFORMATION GIVEN — This message is associated with the STORE FORM statement. It indicates that the form was modified, but the necessary elements in the array were not or vice versa.

Index