Rapid Application Development Tool

Quick Tutorial

Table of Contents

Chapter 1

Getting Started with the Form Painter

Chapter 2

Creating Zooms

Chapter 3

Creating Triggers

Chapter 4

Compiling Generated Code

Chapter 1

Getting Started with the Form Painter

- Form Painter Overview
- Starting the Form Painter
- Using the Form Painter Pull-Down Menus
- Creating a Form Image
- Converting Forms into Input Programs

Form Painter Overview

The Form Painter lets you develop complete data-entry programs written in INFORMIX-4GL. It is an interactive visual front end featuring a full screen editor, a database administration facility, and a screen enhancement builder. The Form Painter acts as the control center for running the *Screen* Code Generator and compilation utility. From within the Form Painter you can:

- Paint a form image, which can be directly converted into an input program.

- Access the database to add, delete, and update tables and columns.

- Store form image information in ASCII files (form specification *.per files), which are compliant with Informix's Perform format and easily moved to other systems.

- Create custom program events that are called from logical *trigger* points within the generated code.

- Copy and move any element of the form image.

- Store form image blocks on a Clipboard.

- Define data-entry areas and how they join with other data entry areas.

- Define how forms work with other forms.

- Specify the order in which input fields are processed on the form.

- Generate default form images with the AutoForm feature.

- Access other programs and tools on the system without leaving the Form Painter.

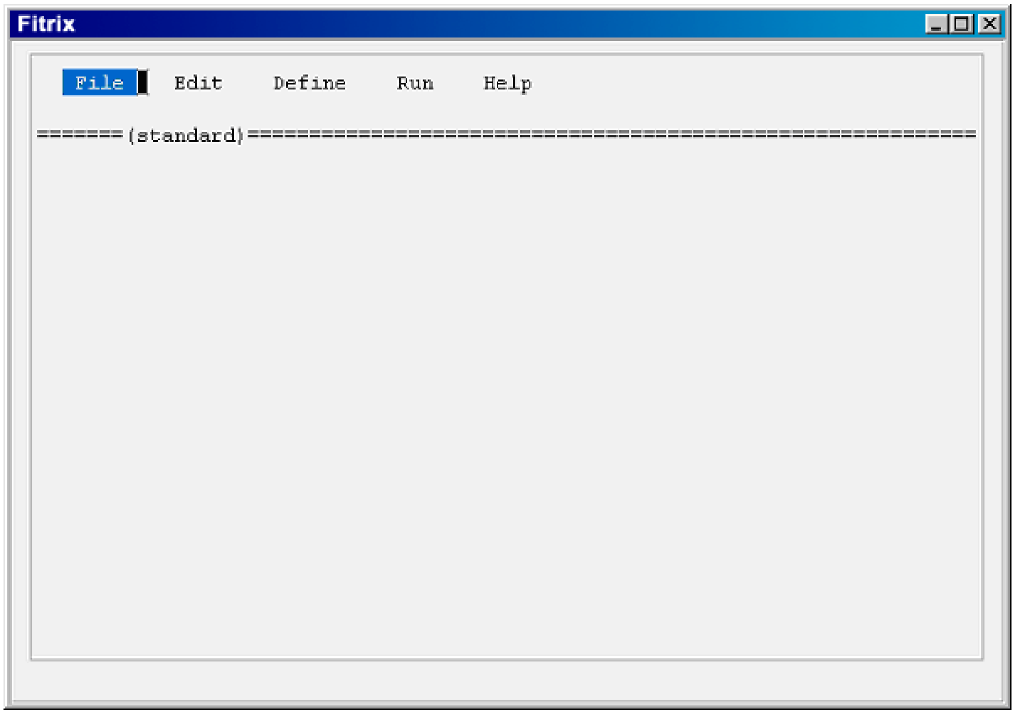
Starting the Form Painter

You can start the Form Painter using the fg.form command. This command has the following syntax:

fg.form –dbname *standard*

Where *standard* is the name for the database you want to use.

After you type this command, the Form Painter loads and displays the following window to your screen:



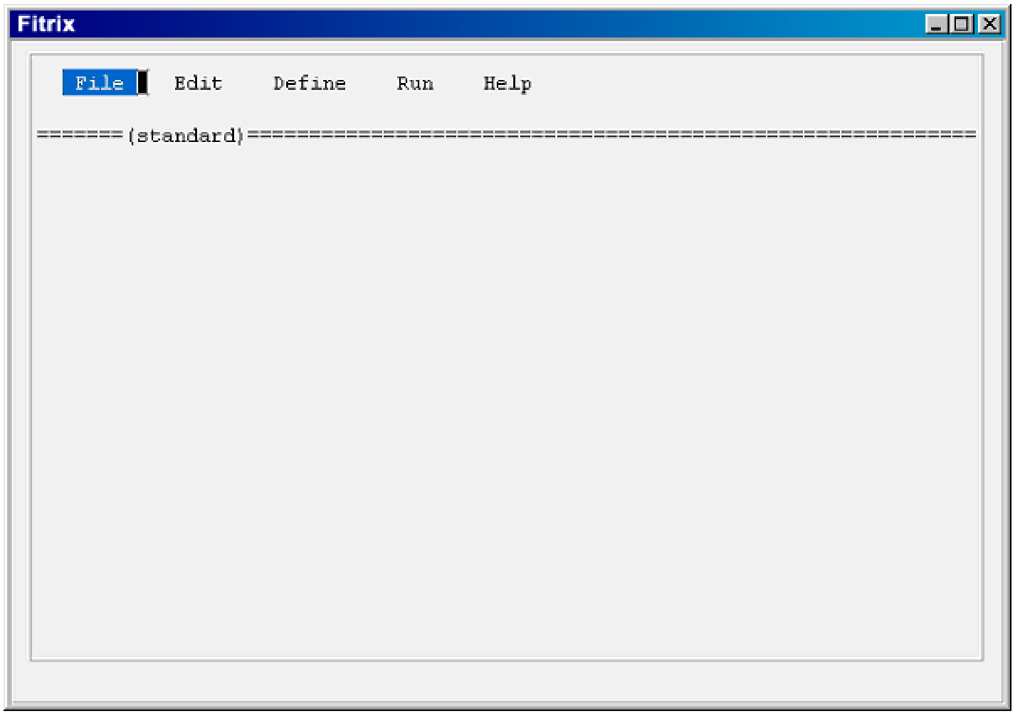
Fitrix
File
Edit
Define
Run
Help
=======(standard)===

You should always start the Form Painter from the directory in which you want to read and write form specification (*.per) files. For example, if you want to write an Accounts Receivable program, you should cd $fg/accounting/ar.4gm.

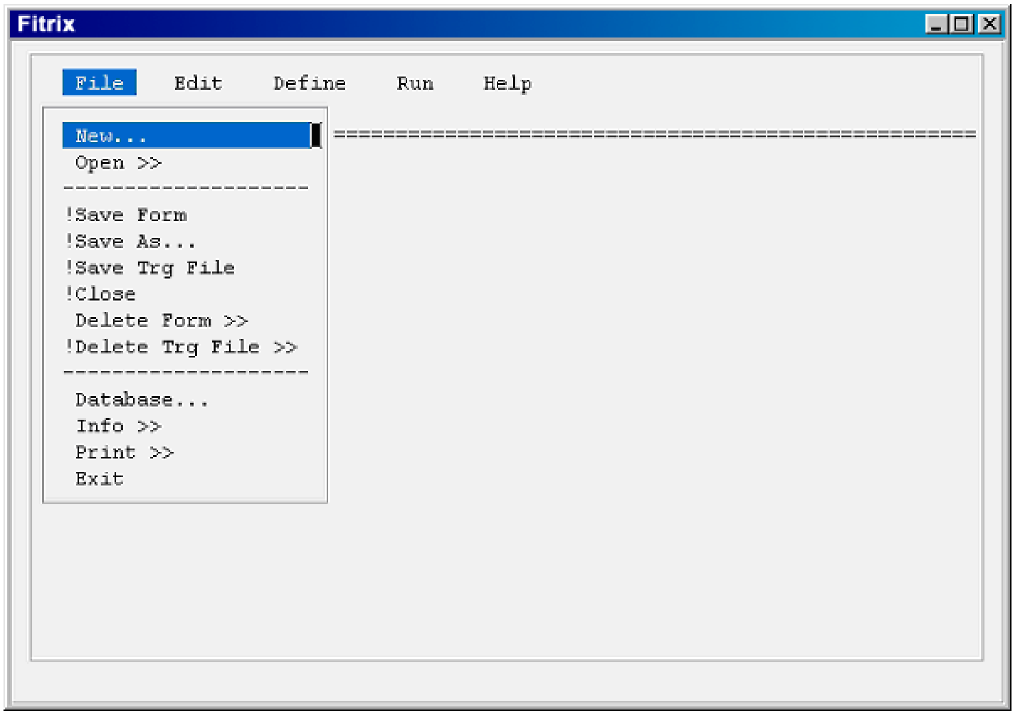
Analogous to generated input programs, the Form Painter consists of two sections: the pull-down menus and the Form Editor.

Using the Form Painter Pull-Down Menus

The Form Painter contains five pull-down menus.


Fitrix
File
Edit
Define
Run
Help
=======(standard)===

You can open a pull-down menu by highlighting it and pressing [ENTER]. You can also open a pull-down menu by typing the first character of the menu name (e.g., type F to open the File pull-down menu). Each pull-down menu contains a number of menu options. You select a menu option by highlighting it and pressing [ENTER].



Options preceded by an exclamation point (!) are not available.
Options followed by greater-than signs (>>) open another menu with additional options.
Options followed by an ellipsis (…) open a subsequent window.

Creating a Form Image

The Form Painter lets you paint form images. You can use the Form Painter to create a new form image or you can open existing form images. A form image graphically represents how your form will look and work once it is built. You paint and edit form images from within the Form Editor. The general steps for creating a new form image are as follows:

1. Select **New** from the File pull-down menu.

2. Enter a name for the new form.

3. Select type you want to use.

In all there are ten screen types you can build. Your main screen is either a header or header/detail screen. The other screens act as secondary screens, some of which you can connect to the main screen.

Screen Type	***Function***
header	Writes to a single database table.
header/detail	Writes to a header table and a detail table.
add-on header	Writes to a peripheral table from the main screen.
add-on detail	Writes to an additional scrolling detail table from the main screen.
extension	Writes to additional columns within the main header table.
zoom	Selects valid values for an input field.
browse	Lists documents in a line-by-line format.
query	Generates a selection prompt for use with report programs.
view-header	Allows you to view data from a peripheral header table.
view-detail	Allows you to view data from a subsequent scrolling detail table.

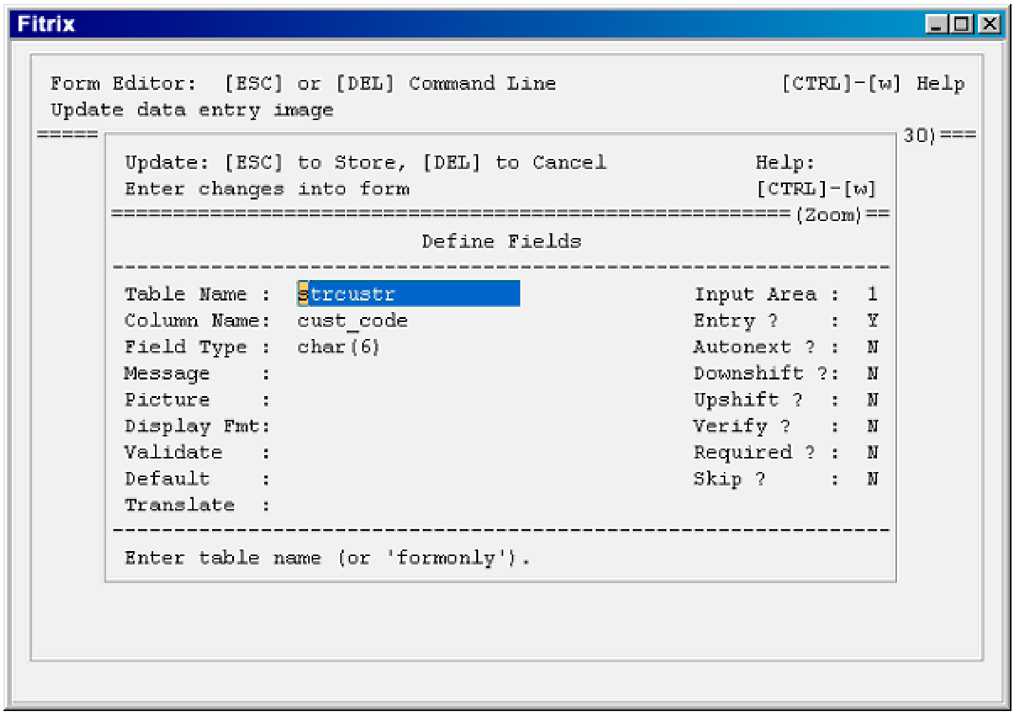
Painting the Form Image

Once you load a form into the Form Editor, you can start painting the form image. Form images contain both text and input field definitions. The Form Editor provides several editing keys.

Keystroke	***Use***
[CTRL]-[a]	Toggles between insert and overstrike mode.
[CTRL]-[x]	Deletes a character.
[CTRL]-[d]	Deletes to the end of a line.
[CTRL]-[u]	Undoes an edit.
[CTRL]-[v]	Marks and cuts a text block to the Clipboard.
[CTRL]-[t]	Cuts a text block and places it on the Clipboard.
[CTRL]-[p]	Pastes a text block.
[F1]	Inserts a blank line above current line.
[F2]	Deletes current line.
[ENTER]	Moves cursor to start of next line.
[HOME]	Moves cursor to top left corner of form.
[	Defines a new field.
]	Lengthens an existing field.
[ESC]	Toggles between pull-down menus and Form Editor.
[DEL]	Returns to pull-down menus.

Defining Fields

When painting the form image, you enter field labels and field attributes. You define a field in the Form Editor by pressing the left bracket ([ ) key. This causes the Define Fields window to appear.



```
Form Editor:  [ESC] or [DEL] Command Line                 [CTRL]-[w] Help
Update data entry image
=====                                                           30)===
      Update: [ESC] to Store, [DEL] to Cancel          Help:
      Enter changes into form                          [CTRL]-[w]
     =======================================================(Zoom)==
                            Define Fields
     ---------------------------------------------------------------
      Table Name :  strcustr                   Input Area :   1
      Column Name:  cust_code                  Entry ?    :   Y
      Field Type :  char(6)                    Autonext ? :   N
      Message    :                             Downshift ?:   N
      Picture    :                             Upshift ?  :   N
      Display Fmt:                             Verify ?   :   N
      Validate   :                             Required ? :   N
      Default    :                             Skip ?     :   N
      Translate  :
     ---------------------------------------------------------------
      Enter table name (or 'formonly').
```

In the Define Fields window you specify the attributes of the field. The attributes are arranged in the window so that the most important and least modified values are supplied first.

Most important are the Table Name and Column Name fields. You can enter values into these two fields directly or use Zoom to select from a list of available values.

The Field Type column is automatically filled in when you enter a valid column name in the Column Name field. You cannot modify the Field Type field because it relates to the column as defined in the database. If you specify Table Name as form only, you are able to specify a value in the Field Type column.

The Input Area field specifies whether the field is on the header (1) or detail (2) part of the form.

The Entry? Field is a Y/N field that determines whether the field is for display purposes only or if it accepts input from the user.

The Message field stores a descriptive line that is displayed when the user positions the cursor in the field.

In the Picture field, you can add a character pattern for displaying the data. For example, area code and phone number fields might use (###) ### - #### as their character pattern.

The Display Fmt field is similar to Display Fmt. It covers the INCLUDE and VALIDATE LIKE Informix attributes. These attributes are also mutually exclusive. Again, refer to your Informix manuals for more information on these attributes.

The Default field lets you set a default value to appear in the field. The user can change default field values.

The Translate field lets you indicate which language you want to use to display data for this field. If specified, translation logic is generated for this field.

The remaining fields are Y/N fields. You can experiment with these fields to see how they affect your input field.

Marking, Copying, and Pasting

When painting your form image, you can cut and paste fields and text. Copying consists of marking a block of text using the arrow keys and selecting the Copy option from the Edit pull-down menu. Once copied, you can paste the text block anywhere in your form image.

To mark and copy a text block:

1. Position the cursor at one corner of the block of text you want to cut.

2. Press [CTRL]-[v] to start the Mark feature

3. Use the arrow keys to highlight the entire block of text you want to mark.

   As you move the cursor, the text you mark appears in reverse video.

4. When you finish marking the entire block, press [CTRL-[v] to copy the text block to the Clipboard

To paste a text block back onto your form image:

1. Position the cursor on the form image where you want the block to appear.

2. Press [CTRL]-[p] to paste the block from the Clipboard to the form image.

3. Use the arrow keys to adjust where you want the block to *stick.*

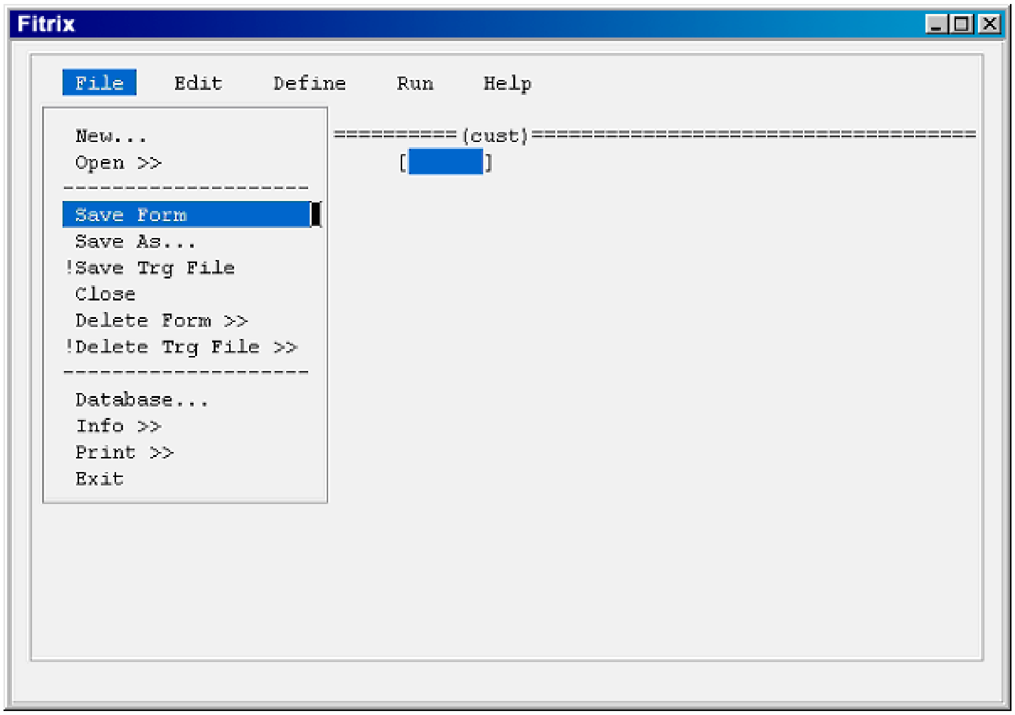
   You can move the entire text block to any location on your form image before you stick it to the image.

4. Pres [ESC] to stick the block to your form image.

In a similar fashion, you can cut a block of text from your form image. Mark the block you want to cut as described above. Once you mark the text block, press [CTRL]-[t] to cut it. You can also paste a cut block back onto your form image in the same manner as described above.

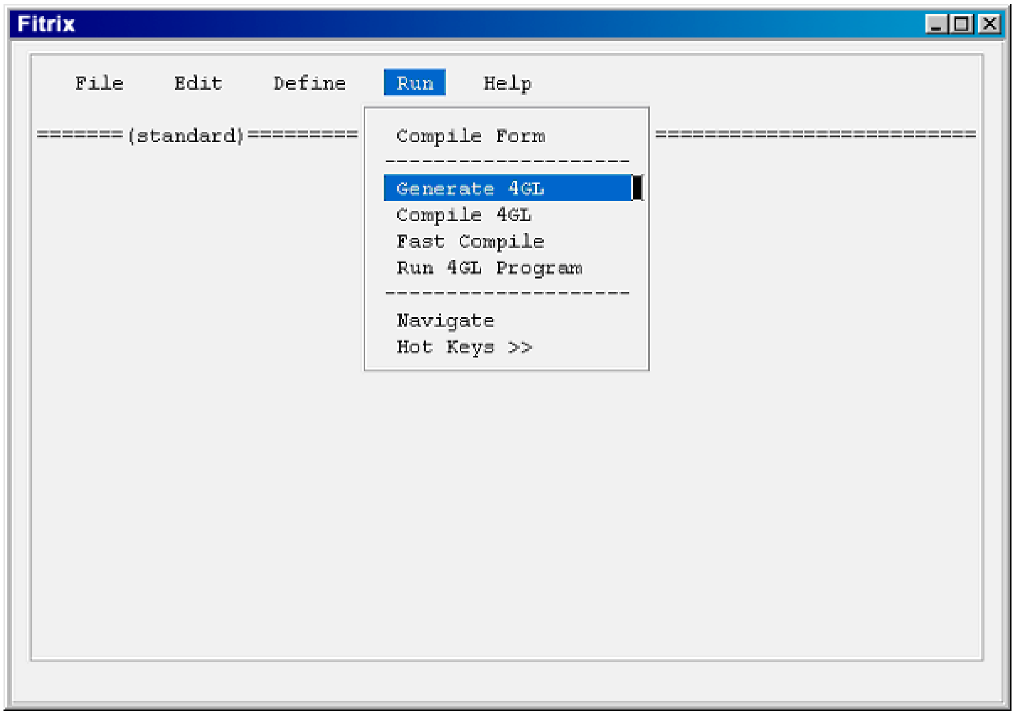
Saving a Form Image

After you paint your form image, you must save it with the Save Form option on the File pull-down menu.



Converting Forms into Input Programs

One you create a form image and save it, you can run the *Screen* Code Generator and compilation utility from within the Form Painter. The Run pull-down menu contains all the options necessary to convert your form into an input program.



In general, you can use the following Run pull-down menu options to convert your form into an input program:

1. Generate 4GL – this option creates the INFORMIX – 4GL source code.

2. Compile 4GL – this option compiles the 4GL code and links in library functions.

3. Run 4GL Program – this option runs the input program in the same manner a user would see it.

Note You can also run the *Screen* Code Generator and compilation utility from outside the Form Painter.

Exercise 1A

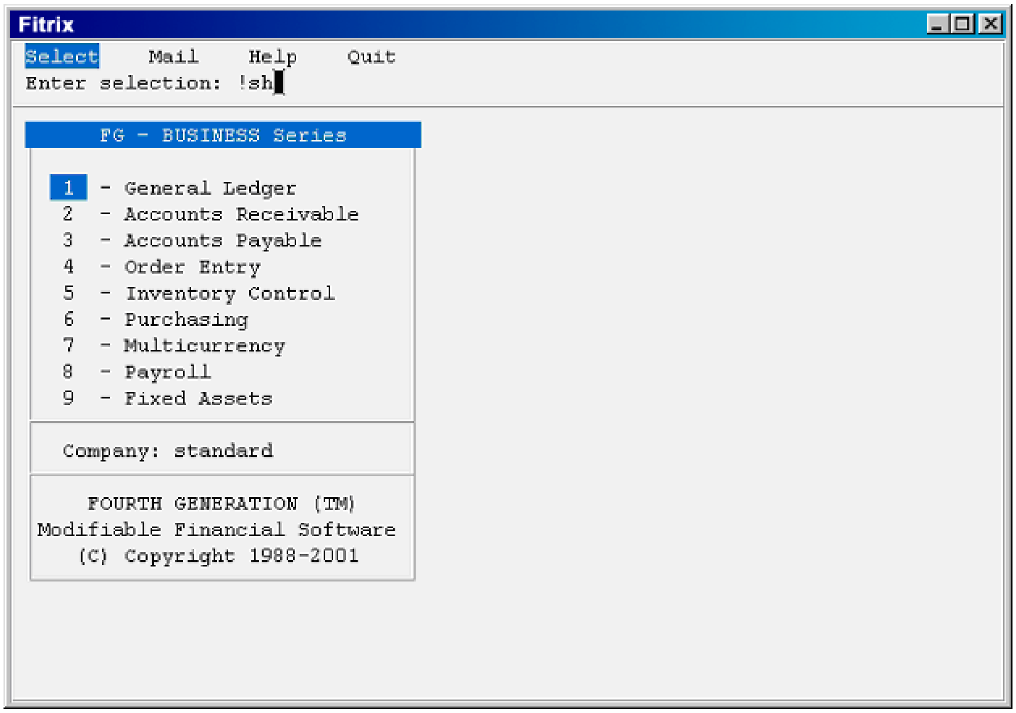
Before you build an input program with the Form Painter, it helps to duplicate this directory structure.

1. Shell out to the Linux Prompt

   (Always use this method when you want to go to a specified program directory described in any of the Exercises that follow)

   Login to the Fitrix Evaluation - Character Mode and shell out using **!sh**

   (To shell out in GUI Mode, from the Menus, select Execute and then Shell)



```
Enter selection: !sh

        FG - BUSINESS Series

   1  - General Ledger
   2  - Accounts Receivable
   3  - Accounts Payable
   4  - Order Entry
   5  - Inventory Control
   6  - Purchasing
   7  - Multicurrency
   8  - Payroll
   9  - Fixed Assets

   Company: standard

     FOURTH GENERATION (TM)
 Modifiable Financial Software
    (C) Copyright 1988-2001
```

2. Once you shell out, go to the following directory:

   cd $fg/accounting/ar.4gm

3. Create a program directory called **i_cust.4gc:**

```
mkdir i_cust.4gc
```

4. Change directory to the i_cust.4gc:

```
cd  i_cust.4gc
```

Once complete, you should be in the i_cust.4gc directory and have the following directory structure:

```
/fitrix/fx_dev/accounting/ar.4gm/i_cust.4gc
```

You can verify the directory you are in, by typing **pwd.**

Exercise 1B

Objective: To start and become familiar with the Form Painter.

Start the Form Painter

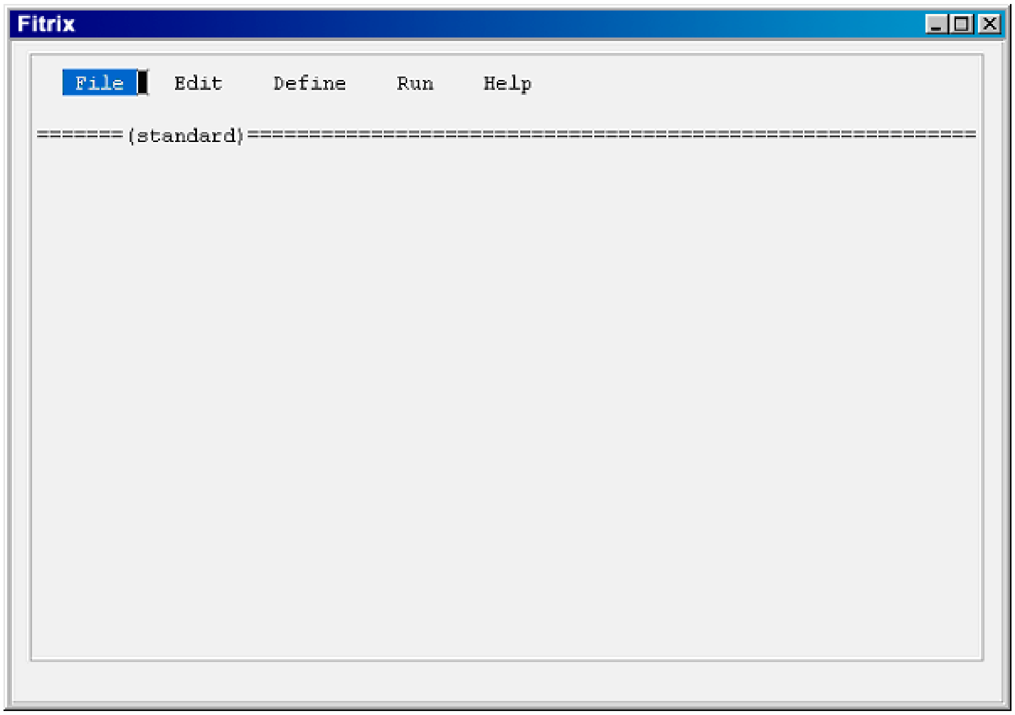
From within the i_cust.4gc directory, you can use the Form Painter to build an input program.

To start the Form Painter, enter:

fg.form

Note The –dbname flag specifies the database you want to use with the Form Painter. If you have been set up to use a different database, specify it in place of standard.

After you enter the fg.form command, the Form Painter appears:


Fitrix
File
Edit
Define
Run
Help
=======(standard)==

The Form Painter lets you design input forms. In the next section you will build a Customer Entry program.

Exercise 1C

Objective: To use the Form Painter to design a Customer Entry form.

There are several steps involved in designing a Customer Entry form.
In general you should use the following sequence:

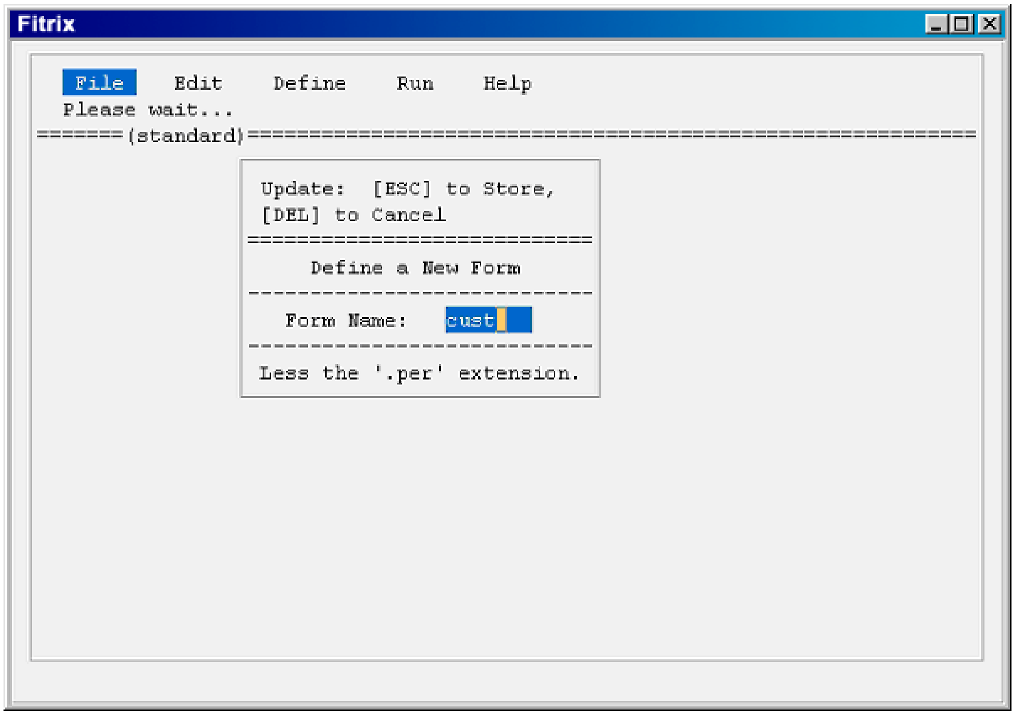
1. Create a new form.
2. Add field labels.
3. Define which table and columns are used.
4. Save the from.

Create Your New Form

The New option on the File pull-down menu lets you create a new form. For this exercise, you will make a header form called cust.

1. Select New from the File pull-down menu.
   The Define a New Form box appears.

2. Enter cust into the Form Name field.



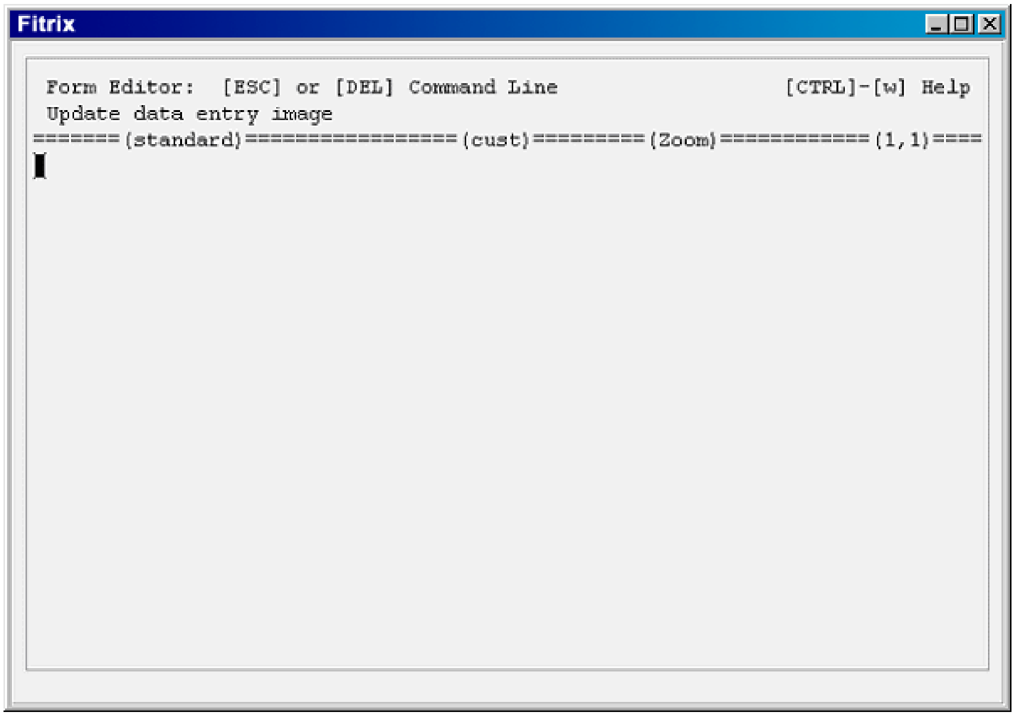
```
 File    Edit    Define    Run    Help
 Please wait...
=======(standard)==============================================================

                     Update:  [ESC] to Store,
                     [DEL] to Cancel
                    ============================
                          Define a New Form
                    ----------------------------
                       Form Name:   cust
                    ----------------------------
                     Less the '.per' extension.
```

The “Select the screen type” box appears.

3. Choose header from the “Select the screen type” box.
   A new form is created and the cursor is placed on the upper left corner of the form (at this point, the form is blank).

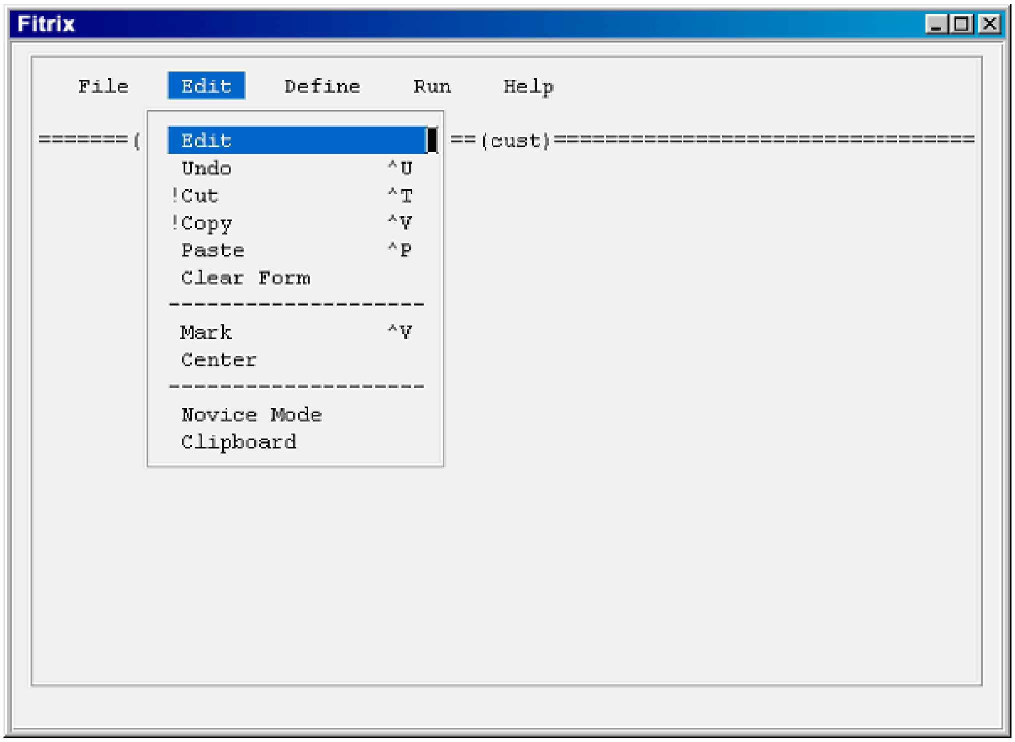
Add Field Labels

Once you create a new form, you can use the Form Editor to add input field labels. If you have just created a new form, your cursor is placed within the Form Editor automatically. The Form Editor lets you enter text and define input fields.



```
 Form Editor:  [ESC] or [DEL] Command Line                  [CTRL]-[w] Help
 Update data entry image
=======(standard)=================(cust)=========(Zoom)============(1,1)====
```

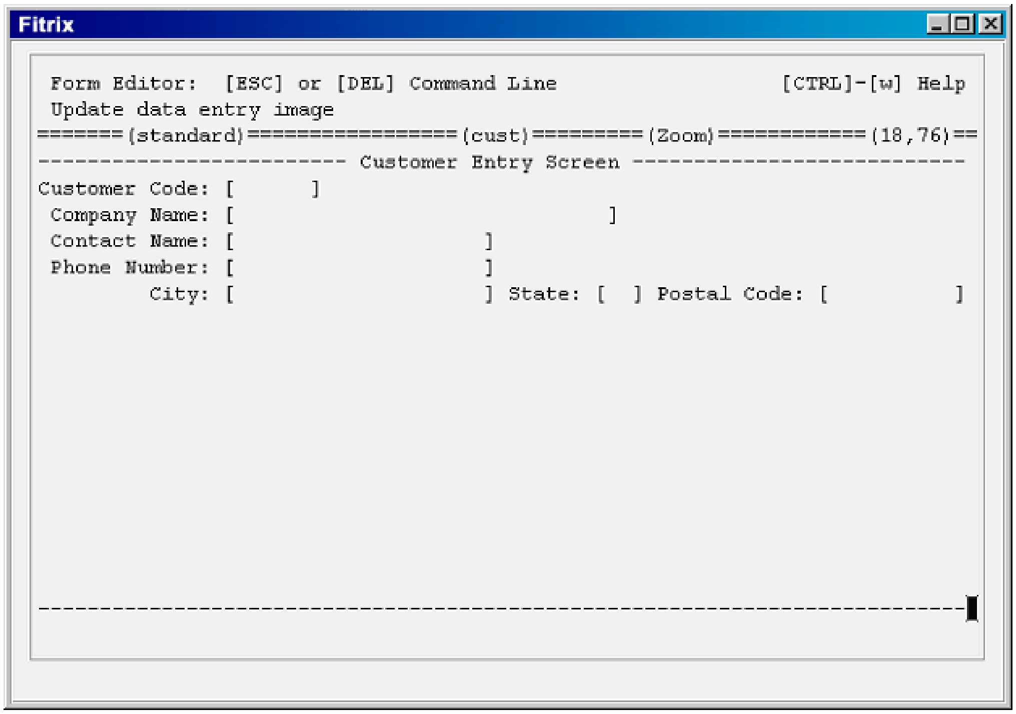
The [ESC] key lets you toggle between the Form Editor and the pull-down menus. You can also move to the Form Editor by selecting Edit from the Edit pull-down menu.



The Form Editor provides a number of useful editing keys and keystrokes to help you design your input form. The following list contains a few of them:

[F1]	Inserts a line.
[F2]	Deletes a line.
[ENTER]	Moves cursor to the start of the next line.
[HOME]	Moves cursor to the upper left corner.
[CTRL]-[a]	Toggles between insert and overstrike mode.
[CTRL]-[x]	Deletes a character.
[CTRL]-[d]	Deletes to the end of a line.
[CTRL]-[u]	Undoes an edit.

For this exercise, use the Form Editor to add input field labels that resemble the following form:



```
 Form Editor:  [ESC] or [DEL] Command Line                  [CTRL]-[w] Help
 Update data entry image
=======(standard)=================(cust)=========(Zoom)============(18,76)==
------------------------- Customer Entry Screen ---------------------------
Customer Code: [      ]
 Company Name: [                              ]
 Contact Name: [                    ]
 Phone Number: [                    ]
         City: [                    ] State: [  ] Postal Code: [          ]










---------------------------------------------------------------------------
```

Make sure to add a dashed line to the bottom of your form. This line will separate your message line from your input form. After you create all the labels, you can define the actual input fields themselves.

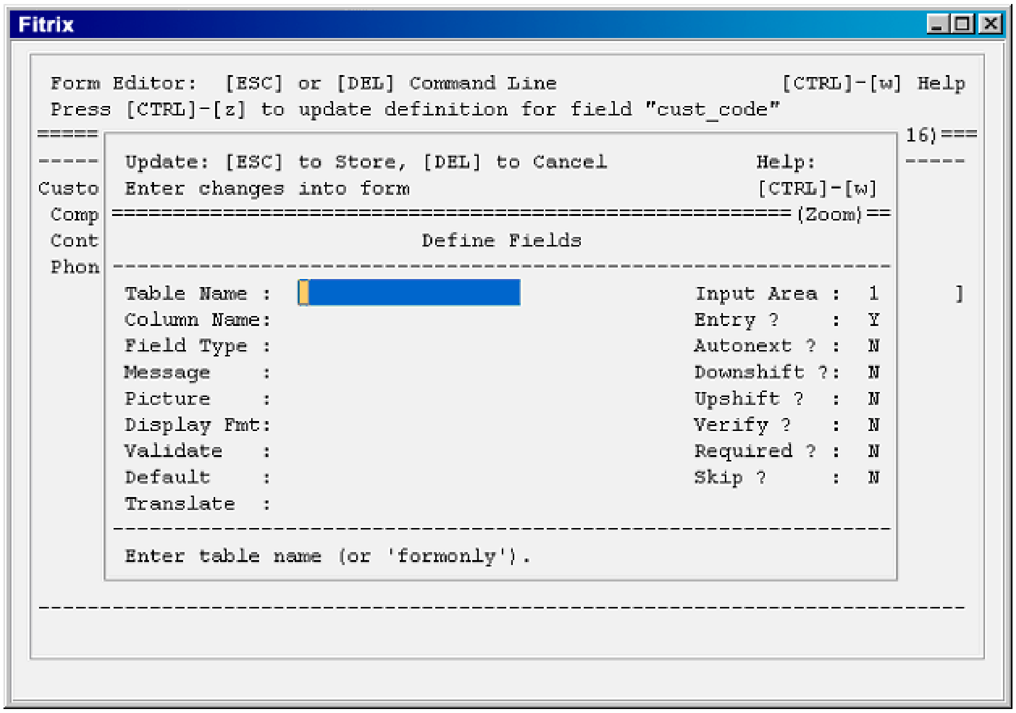
Define Input Fields

At this point, you need to define a corresponding field for each field label on your form. The Form Editor gives you a special key, the left bracket ([ ) key, for defining input fields.

1. Position your cursor to the right of the Customer Code field label you created.

2. Press the left bracket ([ ) key.

   The Define Fields dialog window appears.



```
 Form Editor:  [ESC] or [DEL] Command Line                   [CTRL]-[w] Help
 Press [CTRL]-[z] to update definition for field "cust_code"
=====                                                             16)===
-----  Update: [ESC] to Store, [DEL] to Cancel             Help:  -----
Custo  Enter changes into form                             [CTRL]-[w]
 Comp ======================================================(Zoom)==
 Cont                        Define Fields
 Phon --------------------------------------------------------------
       Table Name :                           Input Area :  1          ]
       Column Name:                           Entry ?    :  Y
       Field Type :                           Autonext ? :  N
       Message    :                           Downshift ?:  N
       Picture    :                           Upshift ?  :  N
       Display Fmt:                           Verify ?   :  N
       Validate   :                           Required ? :  N
       Default    :                           Skip ?     :  N
       Translate  :
      --------------------------------------------------------------
       Enter table name (or 'formonly').

-----------------------------------------------------------------------------
```

Input fields are associated with columns in a database table. They accept data from the user and insert it into a column. In this exercise, each field that you define will correspond to a column in the customer table.

Note — If you see a simplified version of this window, you are in “Novice mode.” For all exercises in this training material, you must be in “Expert mode.” The Edit pull-down menu contains an option that toggles between Expert and Novice mode. When Novice Mode is showing, it means you are in Expert mode and vice versa.

3. Enter **strcustr** in the Table Name field.

4. In the Column Name field, press [CTRL][z].
   A list of all the columns in the **strcustr** table appears.

5. Highlight **cust_code** and press [ESC] to select it.
   Data entered by the user into the Customer Code input field will go directly into this column in the customer database.

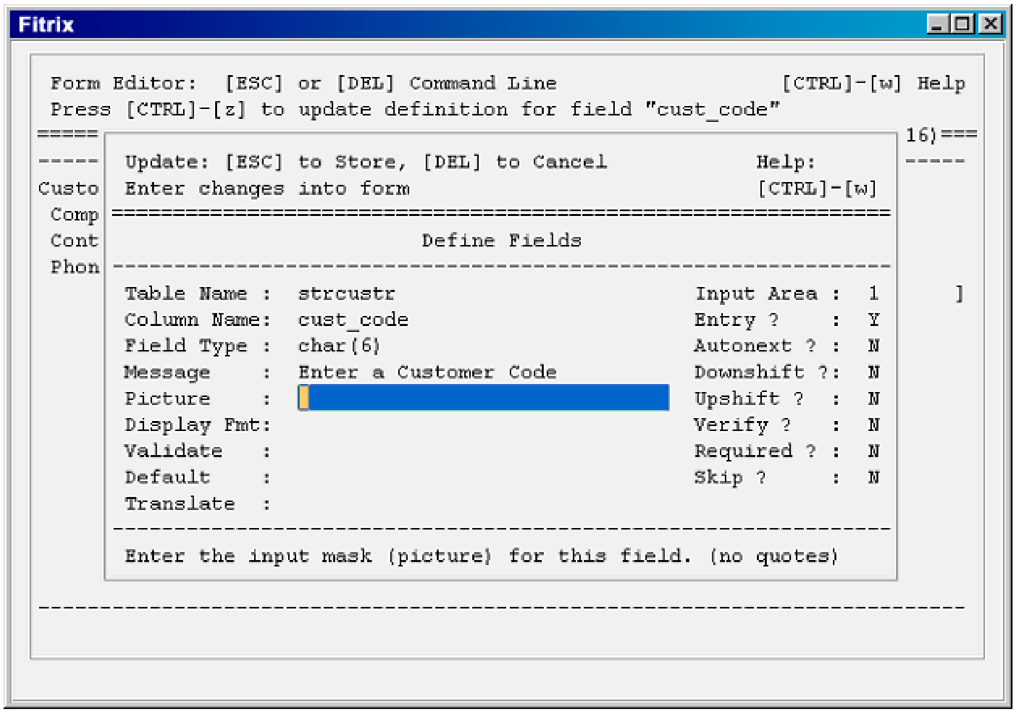
6. Press [ENTER] to move to the Input Area field.
   Notice that when you press [ENTER] the Field Type field gets filled in automatically with the char(6) value.

7. Verify that the Input Area field contains a 1 and press [ENTER].
   For now, all fields will have an Input Area of 1.

8. Accept the Y value for the Entry? Field and press [ENTER].
   A Y value lets the user enter data into this field. An N specifies a no-entry field (i.e., a field in which the user cannot enter data).

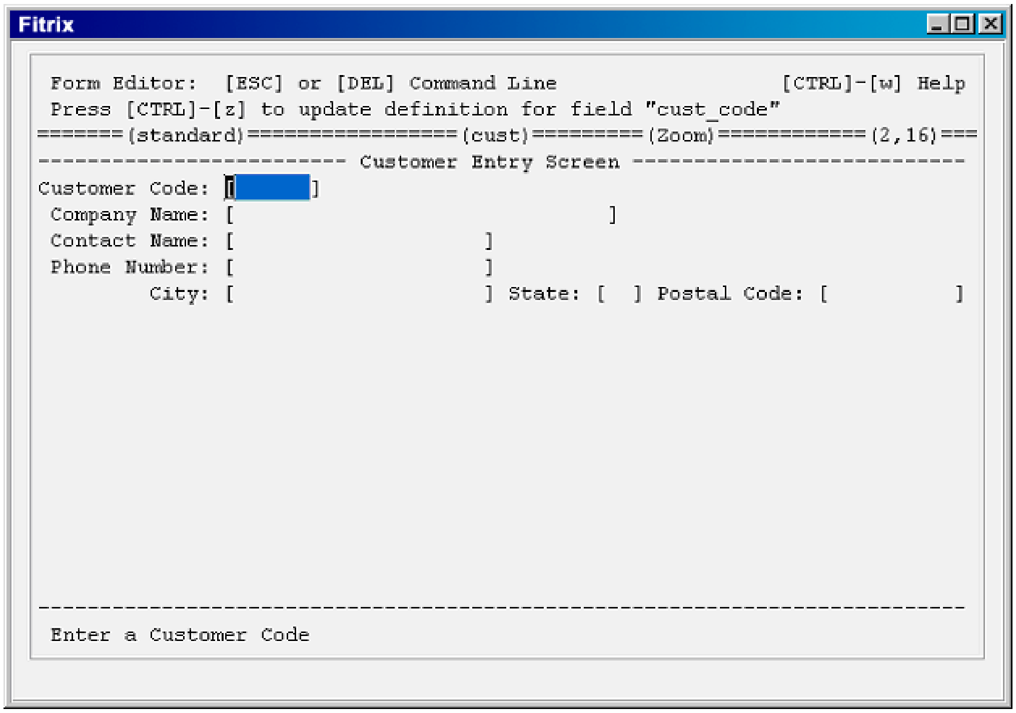
9. Type a message in the Message field and press [ESC].
   This message will appear at the bottom of the form when the cursor is in the Customer Code field.

   For now, you can leave the other fields on the Define Fields window as is. The finished window should appear as follows:



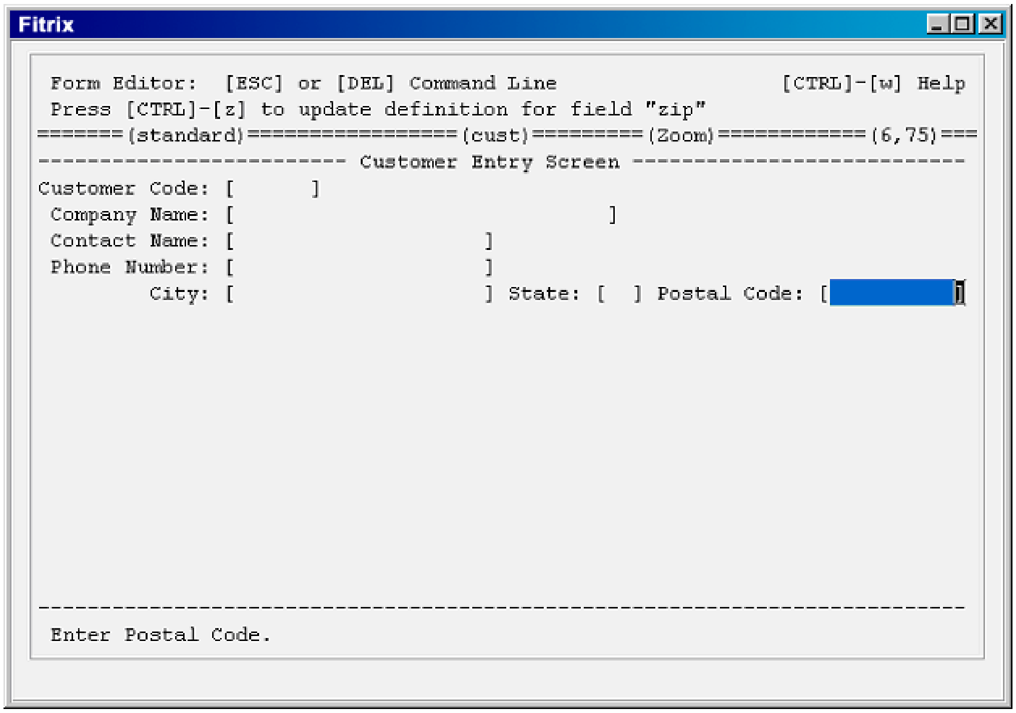
```
Form Editor:  [ESC] or [DEL] Command Line                [CTRL]-[w] Help
Press [CTRL]-[z] to update definition for field "cust_code"
=====                                                           16)===
-----  Update: [ESC] to Store, [DEL] to Cancel            Help:     -----
Custo  Enter changes into form                            [CTRL]-[w]
 Comp  ==============================================================
 Cont                          Define Fields
 Phon  --------------------------------------------------------------
       Table Name :  strcustr                        Input Area :  1     ]
       Column Name:  cust_code                       Entry ?    :  Y
       Field Type :  char(6)                         Autonext ? :  N
       Message    :  Enter a Customer Code           Downshift ?:  N
       Picture    :                                  Upshift ?  :  N
       Display Fmt:                                  Verify ?   :  N
       Validate   :                                  Required ? :  N
       Default    :                                  Skip ?     :  N
       Translate  :
       --------------------------------------------------------------
       Enter the input mask (picture) for this field. (no quotes)
---------------------------------------------------------------------------
```

Once you save the Customer Code field definition, the field appears in the Form Editor as two brackets with a highlight between them. Notice also how the field is automatically sized and the field message appears at the bottom of the screen:



```
 Form Editor:  [ESC] or [DEL] Command Line                [CTRL]-[w] Help
 Press [CTRL]-[z] to update definition for field "cust_code"
=======(standard)=================(cust)=========(Zoom)============(2,16)===
------------------------- Customer Entry Screen --------------------------
Customer Code: [      ]
 Company Name: [                              ]
 Contact Name: [                    ]
 Phone Number: [                    ]
         City: [                    ] State: [  ] Postal Code: [          ]









---------------------------------------------------------------------------
 Enter a Customer Code
```

Follow the same sequence of steps to define the rest of the input fields on your form. When you finish, your form should look as follows:



```
 Form Editor:  [ESC] or [DEL] Command Line                  [CTRL]-[w] Help
 Press [CTRL]-[z] to update definition for field "zip"
=======(standard)=================(cust)=========(Zoom)============(6,75)===
------------------------- Customer Entry Screen ---------------------------
Customer Code: [      ]
 Company Name: [                          ]
 Contact Name: [                    ]
 Phone Number: [                    ]
         City: [                    ] State: [  ] Postal Code: [          ]





------------------------------------------------------------------------------
 Enter Postal Code.
```

After you create a field definition, you might need to re-edit it at some point.

To re-edit a field definition:

1. Place your cursor in the field and press [CTRL]-[z].

   A pop-up menu appears.

2. Select Field from the pop-up menu.

   The Define Fields window appears.

3. Edit the field definition using the Define Fields window and press [ESC] to save your changes.

Save the Form

When you are satisfied with your input form, save it using the Save Form option under the File pull-down menu.

To save a form:

- Select Save Form from the File pull-down menu.

- The Form Painter reads your form image and generates instructions in a form specification (*.per) file. The Screen Code Generator to create source code, which is discussed next, uses this file.

Exercise 1D

Objective: To use the Form Painter to generate, compile, and run your Customer Entry program.

Recall that you can build a demonstration input program from the UNIX command line using fg.sreen, fg.make, and fglrun. The Form Painter gives you the same ability, but you simply select these commands from the Form Painter's Run pull-down menu.

Generate Source Code

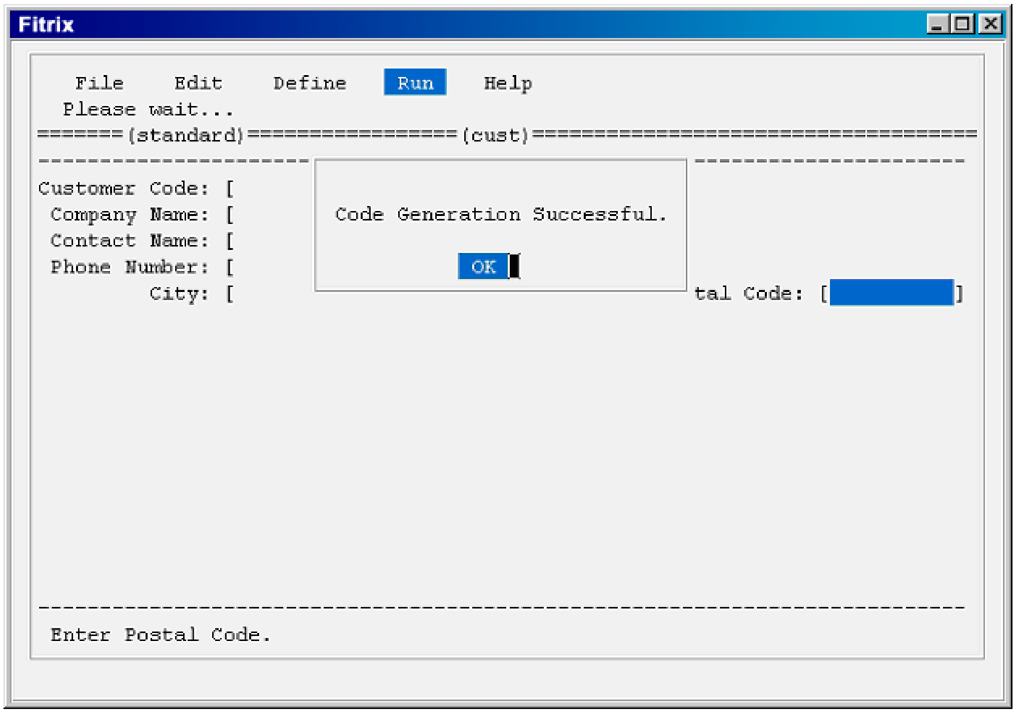
1. Select Generate 4GL from the Run pull-down menu.
   A pop-up menu appears asking you which forms to generate code for.

2. Select All Forms from the pop-up menu.
   A message box appears asking you if you want to generate code for local forms only.

3. Select YES on the "Local forms only" message box

   The *Screen* Code Generator is run and this code scrolls past your screen as it is created based on your cust form. You might see a message indicating that your cust form is not current. If this happens, simply select YES from the message box.

   When the *Screen* Code Generator finishes, the following message appears:



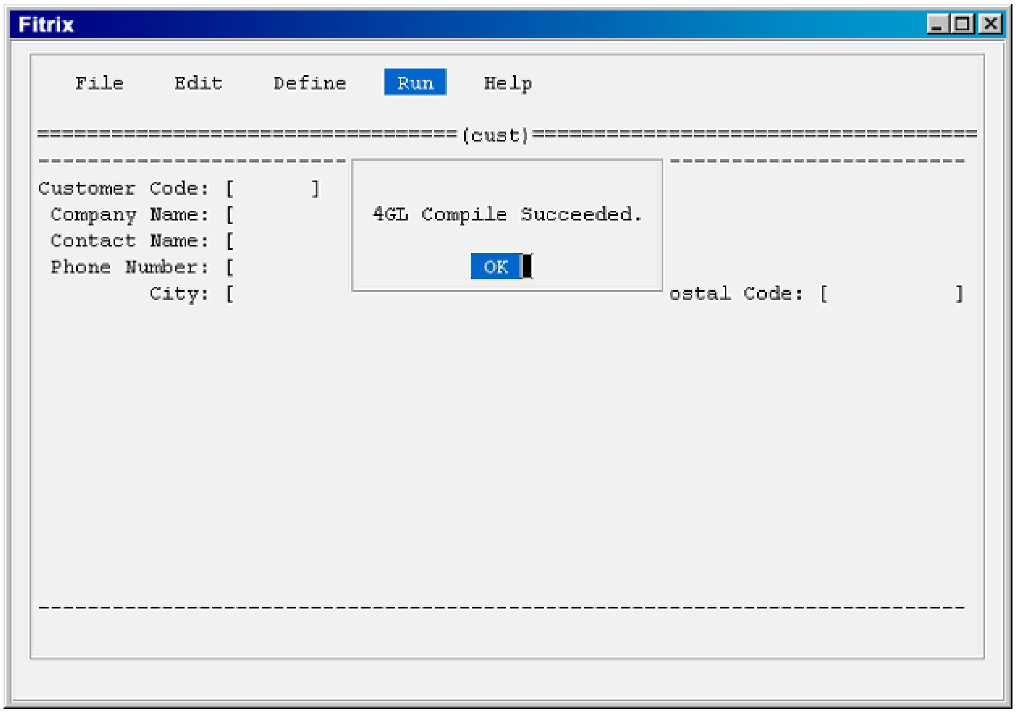
```
 Please wait...
=======(standard)=================(cust)====================================
----------------------                               ----------------------
Customer Code: [
 Company Name: [           Code Generation Successful.
 Contact Name: [
 Phone Number: [                       OK
         City: [                                      tal Code: [          ]

-----------------------------------------------------------------------------
 Enter Postal Code.
```

4. Press "Enter" to acknowledge.

Compile the Code

- Select Compile 4GL from the Run pull-down menu.

  The Form Painter calls the compilation utility and creates a program file. When done, the following message appears:

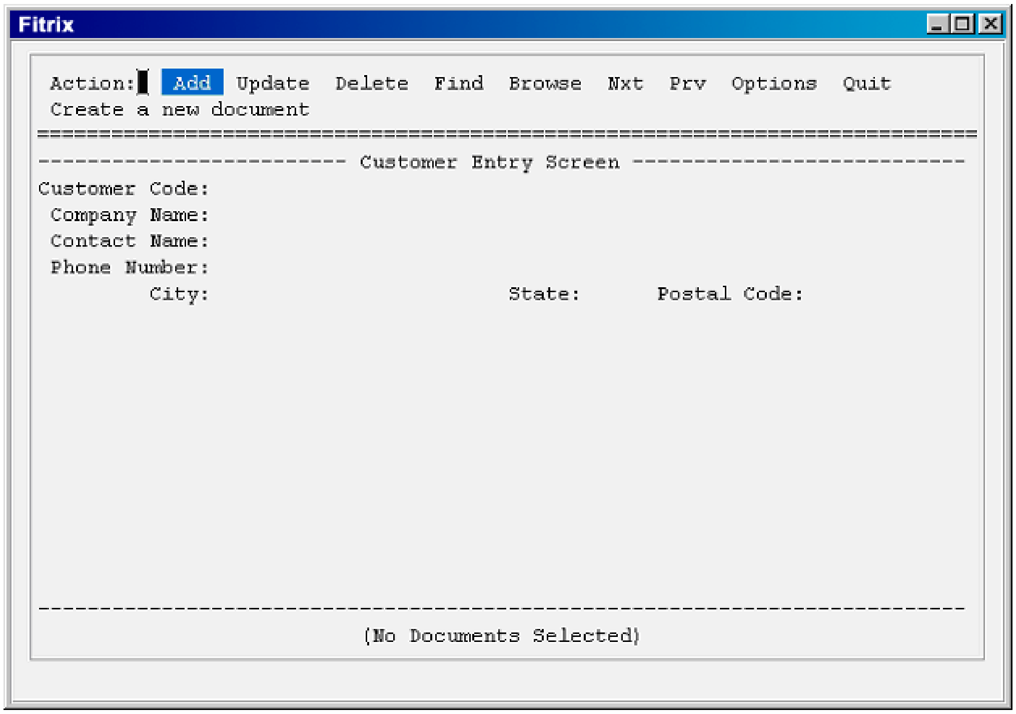


```
=================================(cust)===================================
------------------------                         -----------------------
Customer Code: [      ]
 Company Name: [            4GL Compile Succeeded.
 Contact Name: [
 Phone Number: [
         City: [                                 ostal Code: [          ]

--------------------------------------------------------------------------
```

Press “Enter” to acknowledge.

Run Your Customer Entry Program

1. Select Run 4GL Program from the Run pull-down menu.
   The Form Painter runs your Customer Entry program.



```
Fitrix
Action: Add  Update  Delete  Find  Browse  Nxt  Prv  Options  Quit
Create a new document
============================================================================
------------------------- Customer Entry Screen ---------------------------
Customer Code:
 Company Name:
 Contact Name:
 Phone Number:
         City:                        State:      Postal Code:

---------------------------------------------------------------------------
                       (No Documents Selected)
```

2. Use the ring menu to “test drive” your input program. When you finish, select Quit to return to the Form Painter.

Chapter 2

Creating Zooms

- Zoom Screen Overview
- Painting a Zoom Image
- Attaching the Zoom Screen

Zoom Screen Overview

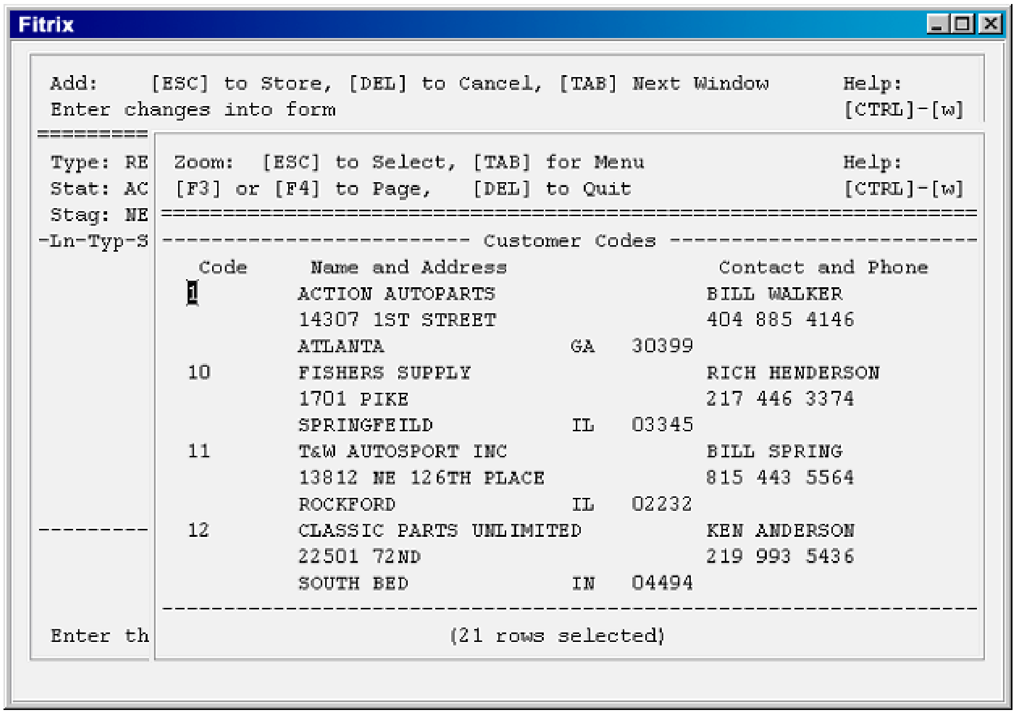
A Zoom is a data validation feature that shows the user a list of valid values for an input field. Zooms are created from zoom screen types. When users initiate a Zoom, they can enter selection criteria on the fields in the Zoom.
The Zoom then takes the selection criteria and returns all valid values that meet the criteria. Users can select the value they want to use from the values the Zoom returns.

Zooms make the data-entry process much more accurate and efficient. Field values are validated before they get inserted. In general, creating Zooms is a two-step process:

1. **Paint and define the zoom screen image.**
2. **Attach the zoom screen to a field on your main input screen**.

Painting a Zoom Image

You define Zooms by using the Form Painter to paint their image. Once you paint the image of the zoom screen, you must also specify from which field on your main input form the zoom screen can be activated. For example, the following application has a zoom screen attached to the Customer Code field.



```
 Add:    [ESC] to Store, [DEL] to Cancel, [TAB] Next Window      Help:
 Enter changes into form                                         [CTRL]-[w]
=========
 Type: RE  Zoom:  [ESC] to Select, [TAB] for Menu                Help:
 Stat: AC  [F3] or [F4] to Page,   [DEL] to Quit                 [CTRL]-[w]
 Stag: NE ===================================================================
-Ln-Typ-S ------------------------- Customer Codes --------------------------
             Code     Name and Address                 Contact and Phone
            1        ACTION AUTOPARTS                 BILL WALKER
                     14307 1ST STREET                 404 885 4146
                     ATLANTA               GA   30399
            10       FISHERS SUPPLY                   RICH HENDERSON
                     1701 PIKE                        217 446 3374
                     SPRINGFEILD           IL   03345
            11       T&W AUTOSPORT INC                BILL SPRING
                     13812 NE 126TH PLACE             815 443 5564
                     ROCKFORD              IL   02232
---------   12       CLASSIC PARTS UNLIMITED          KEN ANDERSON
                     22501 72ND                       219 993 5436
                     SOUTH BED             IN   04494
          -------------------------------------------------------------------
 Enter th                        (21 rows selected)
```

To define a Zoom:

1. Select New from the File pull-down menu.

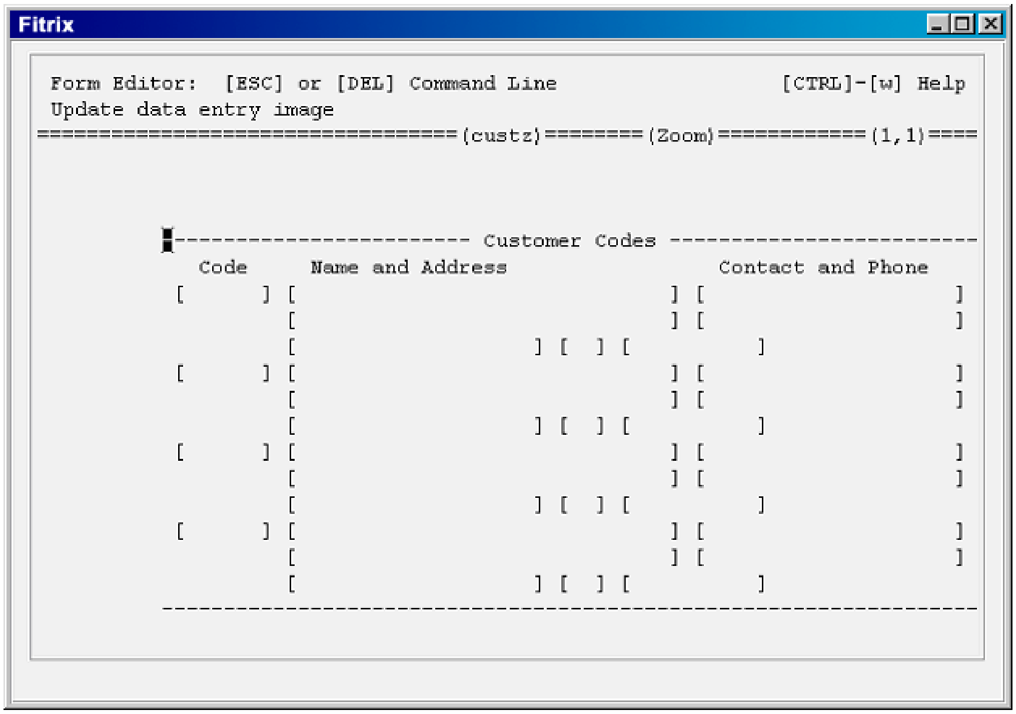
2. Specify a name for the zoom screen.

   Traditionally, zoom screens are given a name that includes the letters z, such as custz, stockz, etc.

3. Select zoom as the screen type.

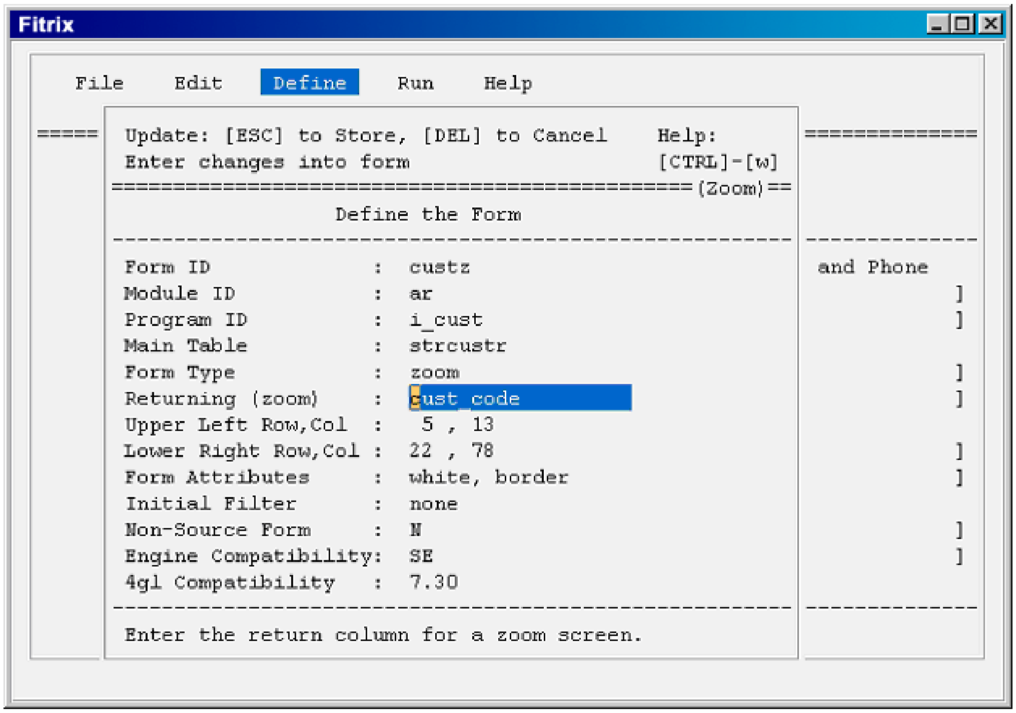
4. Use the Form Painter to paint and save the zoom image.

   Because zoom screens usually contain several rows of duplicate field definitions, use mark, copy, and paste to speed your creation of the zoom image.



```
 Form Editor:  [ESC] or [DEL] Command Line                    [CTRL]-[w] Help
 Update data entry image
===================================(custz)========(Zoom)============(1,1)====



          -------------------------- Customer Codes --------------------------
             Code     Name and Address                  Contact and Phone
           [      ] [                               ] [                    ]
                    [                               ] [                    ]
                    [                    ] [  ] [          ]
           [      ] [                               ] [                    ]
                    [                               ] [                    ]
                    [                    ] [  ] [          ]
           [      ] [                               ] [                    ]
                    [                               ] [                    ]
                    [                    ] [  ] [          ]
           [      ] [                               ] [                    ]
                    [                               ] [                    ]
                    [                    ] [  ] [          ]
          --------------------------------------------------------------------
```

After you paint and save your zoom image, you need to use the Form Defaults option on the Define pull-down menu. The Form Defaults option opens the Define the Form window. This window lets you specify from which field the zoom screen can be activated.



```
=====  Update: [ESC] to Store, [DEL] to Cancel    Help:     ==============
       Enter changes into form                      [CTRL]-[w]
       ==============================================(Zoom)==
                            Define the Form
       ------------------------------------------------------- --------------
       Form ID               :   custz                           and Phone
       Module ID             :   ar                                        ]
       Program ID            :   i_cust                                    ]
       Main Table            :   strcustr
       Form Type             :   zoom                                      ]
       Returning (zoom)      :   cust_code                                 ]
       Upper Left Row,Col    :    5 , 13
       Lower Right Row,Col   :   22 , 78                                   ]
       Form Attributes       :   white, border                             ]
       Initial Filter        :   none
       Non-Source Form       :   N                                         ]
       Engine Compatibility:     SE                                        ]
       4gl Compatibility     :   7.30
       ------------------------------------------------------- --------------
       Enter the return column for a zoom screen.
```

Make sure to specify a value in the Returning (zoom) field. This field specifies where the returning value gets placed. In most cases, this is the field in which you attach the Zoom. If you are not sure of the field, press [CTRL]-[z] to see a list of available fields.

Attaching the Zoom Screen

You can attach a zoom screen to the main screen of your program using the Form Painter.

To attach a zoom screen to an input field:

1. Open the form that contains the field that you want to attach the zoom screen to.

   In most cases, you attach zoom screens to header or header/detail screens, but this is not necessarily the case.

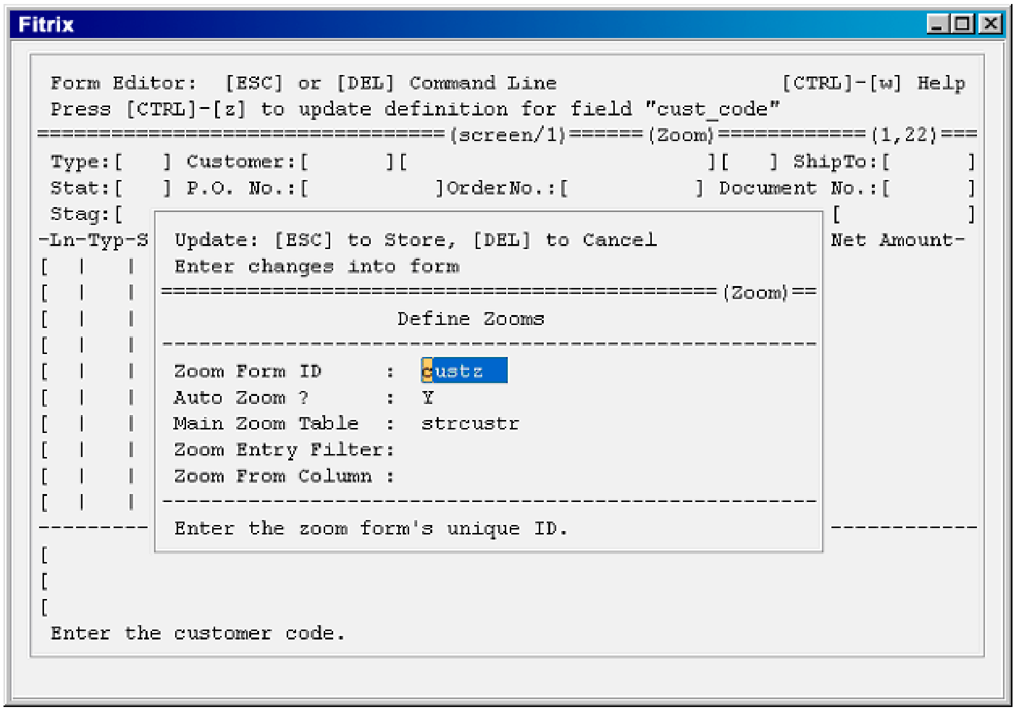
2. Highlight the field you want to attach the zoom screen to.

3. Press [CTRL]-[z]

   Note the irony here. You activate a Form Painter Zoom in order to define a Zoom for your input program. When you press [CTRL]-[z] a pop-up menu appears that contains all the items available for you to attach to the input field.

4. Select Zoom… from the list

   The Define Zooms window appears.



```
 Form Editor:  [ESC] or [DEL] Command Line                  [CTRL]-[w] Help
 Press [CTRL]-[z] to update definition for field "cust_code"
=================================(screen/1)======(Zoom)============(1,22)===
 Type:[   ] Customer:[      ][                        ][   ] ShipTo:[      ]
 Stat:[   ] P.O. No.:[          ]OrderNo.:[          ] Document No.:[      ]
 Stag:[                                                      [             ]
-Ln-Typ-S  Update: [ESC] to Store, [DEL] to Cancel          Net Amount-
[   |   |  Enter changes into form
[   |   | ============================================(Zoom)==
[   |   |                Define Zooms
[   |   | ----------------------------------------------------
[   |   |  Zoom Form ID      :  custz
[   |   |  Auto Zoom ?       :  Y
[   |   |  Main Zoom Table   :  strcustr
[   |   |  Zoom Entry Filter:
[   |   |  Zoom From Column :
[   |   | ----------------------------------------------------
---------  Enter the zoom form's unique ID.                 ------------
[
[
[
 Enter the customer code.
```

5. Fill in the Define Zooms window and press [ESC].

   The Define Zooms window lets you specify how you want the zoom screen to be attached.

The Define Zooms window contains several fields. Perhaps the Zoom Form ID field is most important. In this field, you place the name of your Zoom screen. You should make sure that the Main Zoom Table field contains the correct value. If you want to add AutoZoom capability, specify Y in the AutoZoom field.

The Zoom Entry Filter field lets you assign a selection filter to the Zoom. The last field, Zoom From Column, lets you specify a table and column name for the Zoom if they differ from the column on the main screen.

Exercise 2A

Objective: To add terms_code to the i_cust.4gc program.

Start the Form Painter

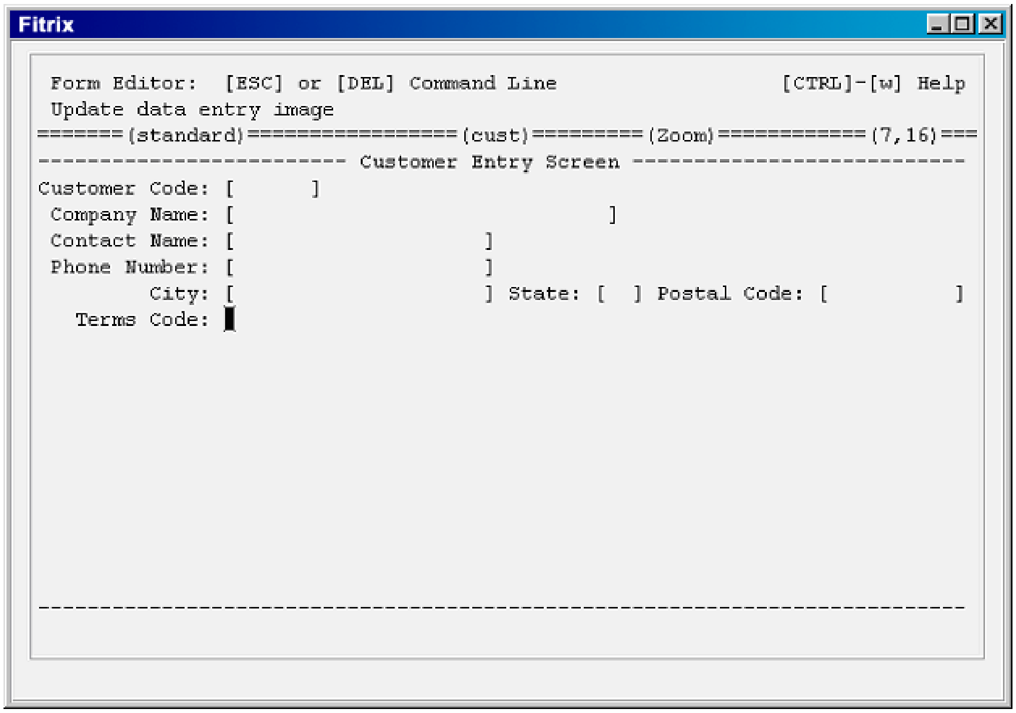
1. Change Directory to $fg/accounting/ar.4gm/i_cust.4gc

2. Start the Form Painter.

Add Terms Code Field to Your Screen

1. Select Open from the File pull-down menu.
   The Form Painter opens your cust.per file. If you have additional form specification (*.per) files in this directory, you have to select cust from the list.

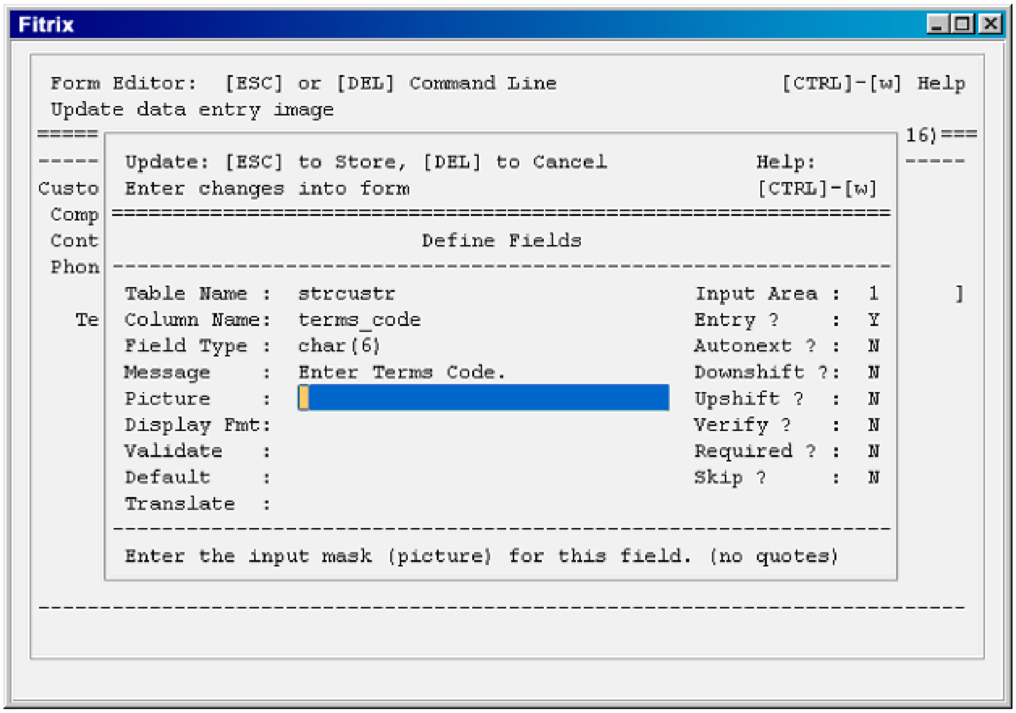
2. Add Terms Code field label in the bottom half of your screen.



```
 Form Editor:  [ESC] or [DEL] Command Line                  [CTRL]-[w] Help
 Update data entry image
=======(standard)=================(cust)=========(Zoom)============(7,16)===
------------------------- Customer Entry Screen ---------------------------
Customer Code: [      ]
 Company Name: [                             ]
 Contact Name: [                   ]
 Phone Number: [                   ]
         City: [                   ] State: [  ] Postal Code: [          ]
   Terms Code: ▌











---------------------------------------------------------------------------
```

3. Define the Terms Code field by pressing a left bracket [after the field.
   The Define Fields window appears.

4. Define the Terms Code field using the values shown below, then press [ESC] to save the definition.



```
Form Editor:  [ESC] or [DEL] Command Line                   [CTRL]-[w] Help
Update data entry image
=====                                                        16)===
-----  Update: [ESC] to Store, [DEL] to Cancel          Help:  -----
Custo  Enter changes into form                          [CTRL]-[w]
 Comp  ===============================================================
 Cont                        Define Fields
 Phon  ---------------------------------------------------------------
       Table Name :  strcustr                     Input Area :  1      ]
   Te  Column Name:  terms_code                   Entry ?    :  Y
       Field Type :  char(6)                      Autonext ? :  N
       Message    :  Enter Terms Code.            Downshift ?:  N
       Picture    :                               Upshift ?  :  N
       Display Fmt:                               Verify ?   :  N
       Validate   :                               Required ? :  N
       Default    :                               Skip ?     :  N
       Translate  :
       ---------------------------------------------------------------
       Enter the input mask (picture) for this field. (no quotes)
---------------------------------------------------------------------------
```

Save, Generate, and Compile

1. Select Save Form from the File pull-down menu.

2. Select Generate 4GL from the Run pull-down menu.

   When you Generate code, please select Option 1 if a message similar to the one given below appears.

   The file globals.org already exists!
   Would you like to:

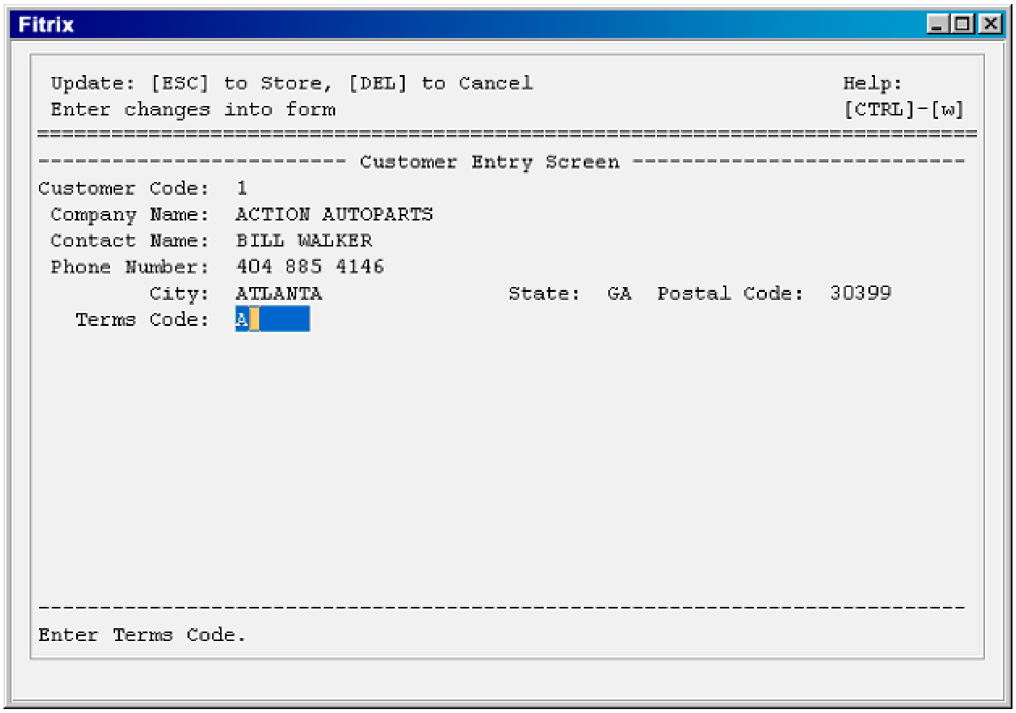
   1) Overwrite globals.org
   2) …..
   3) ……
   4)

3. Select Compile 4GL from the Run pull-down menu.

Run Your Customer Entry Program

1. Select Run 4GL Program from the Run pull-down menu.

2. Use Find to select an existing customer and Update Terms code for that customer.



```
 Update: [ESC] to Store, [DEL] to Cancel                           Help:
 Enter changes into form                                           [CTRL]-[w]
=============================================================================
------------------------- Customer Entry Screen ---------------------------
Customer Code:   1
 Company Name:   ACTION AUTOPARTS
 Contact Name:   BILL WALKER
 Phone Number:   404 885 4146
         City:   ATLANTA                State:   GA   Postal Code:   30399
   Terms Code:   A

---------------------------------------------------------------------------
Enter Terms Code.
```

3. When finished, quit the Customer Entry program and the Form Painter.

Exercise 2B

Objective: To create a zoom screen so users can select from a reference list of Terms codes.

Create Zoom Screen

1. Start the Form Painter.
2. Select New from the File pull-down menu.
   The Define a New Form box appears.
3. Name the new form termz
   The Select a Screen Type box appears.
4. Use the down arrow to view the screen type list and select zoom as the screen type.

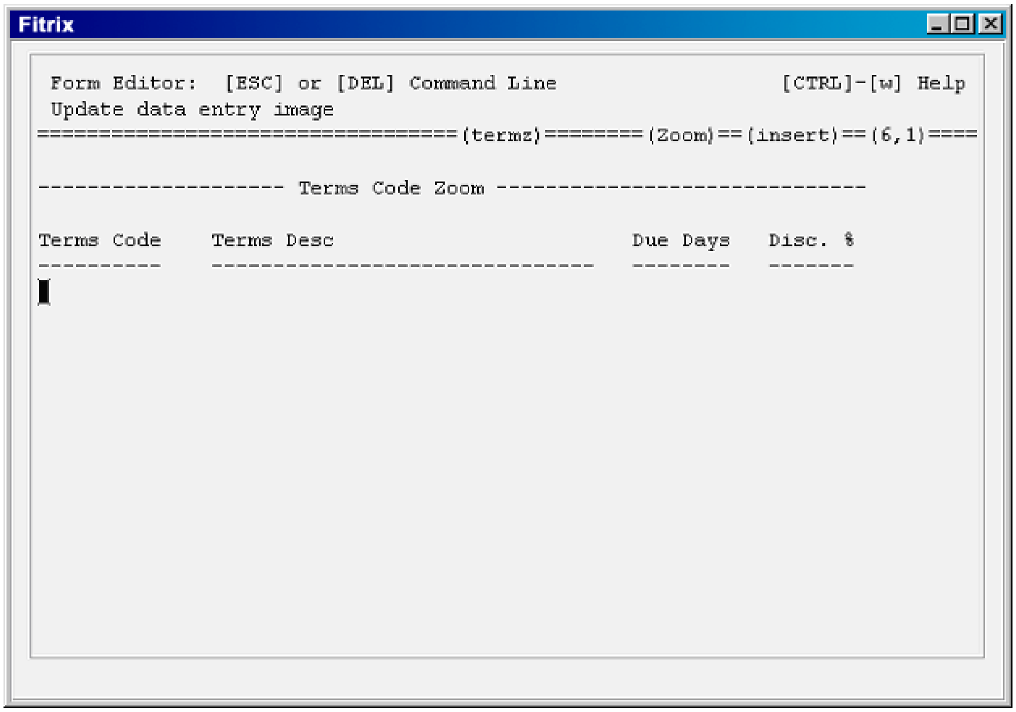
Create a Title

Enter a title for the zoom screen, such as:

____________________Terms Code Zoom____________________

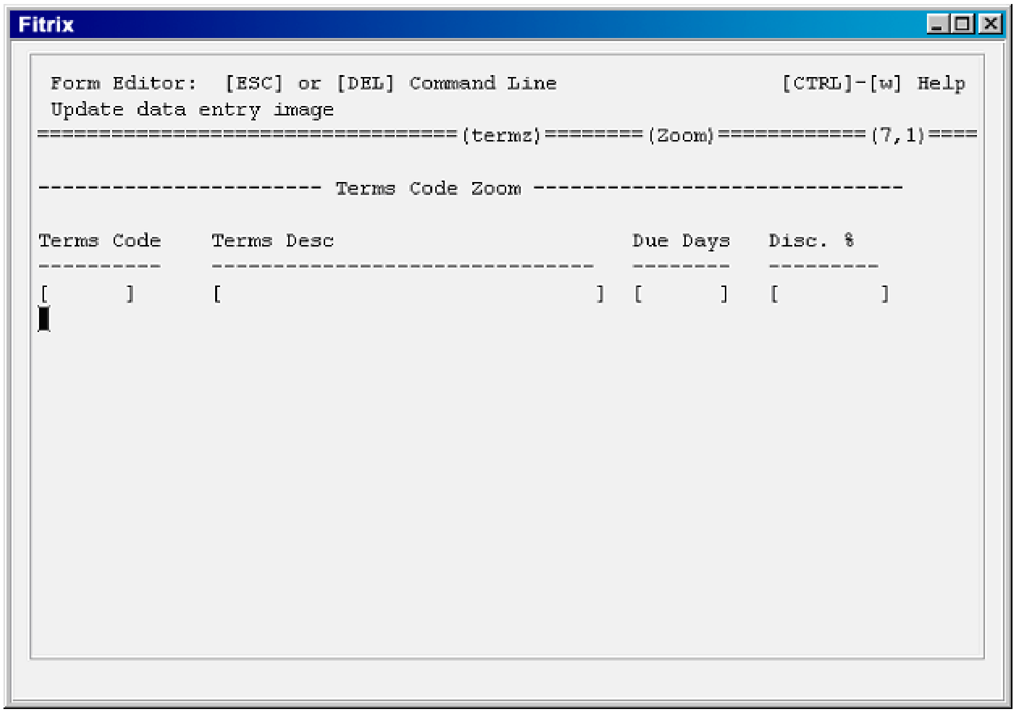
Create the Column Headings

1. Create the column headings for the Zoom.
   A zoom screen displays data in a row-by-row format.



```
 Form Editor:  [ESC] or [DEL] Command Line                [CTRL]-[w] Help
 Update data entry image
==================================(termz)========(Zoom)==(insert)==(6,1)====

-------------------- Terms Code Zoom ------------------------------

Terms Code    Terms Desc                        Due Days   Disc. %
----------    -------------------------------   --------   -------
```

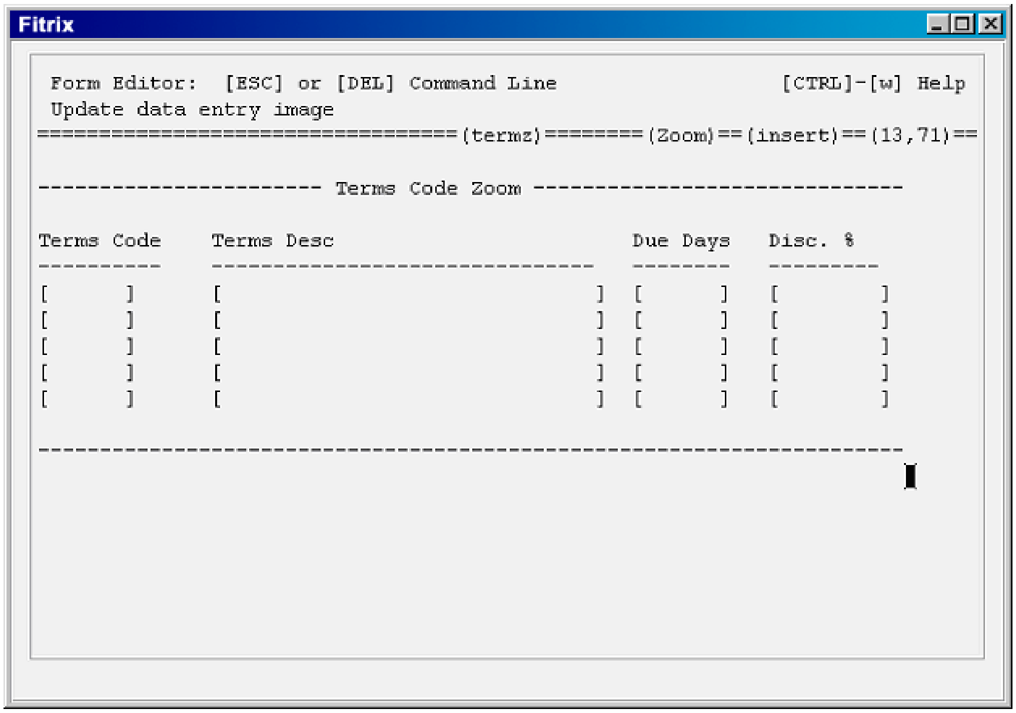
2. Add field definitions using the columns in the Terms Definition Table (strtermr). (terms_code, terms_desc, due_days and disc_pct)



```
 Form Editor:  [ESC] or [DEL] Command Line                 [CTRL]-[w] Help
 Update data entry image
=================================(termz)========(Zoom)============(7,1)====

----------------------- Terms Code Zoom ------------------------------

Terms Code    Terms Desc                          Due Days   Disc. %
----------    -------------------------------     --------   ---------
[      ]      [                              ]  [      ]   [        ]
```

3. Use the Mark (Ctrl-V), Copy(Highlight the entire line with the cursor), and Paste (Ctrl-P and Enter on a new line) options to add four more rows of field definitions.

   Your finished zoom screen should look as follows:



```
 Form Editor:  [ESC] or [DEL] Command Line                  [CTRL]-[w] Help
 Update data entry image
==================================(termz)========(Zoom)==(insert)==(13,71)==

----------------------- Terms Code Zoom ------------------------------

Terms Code    Terms Desc                         Due Days   Disc. %
----------    -------------------------------    --------   ---------
[      ]      [                              ]   [      ]   [        ]
[      ]      [                              ]   [      ]   [        ]
[      ]      [                              ]   [      ]   [        ]
[      ]      [                              ]   [      ]   [        ]
[      ]      [                              ]   [      ]   [        ]

----------------------------------------------------------------------
```

Specify Form Defaults

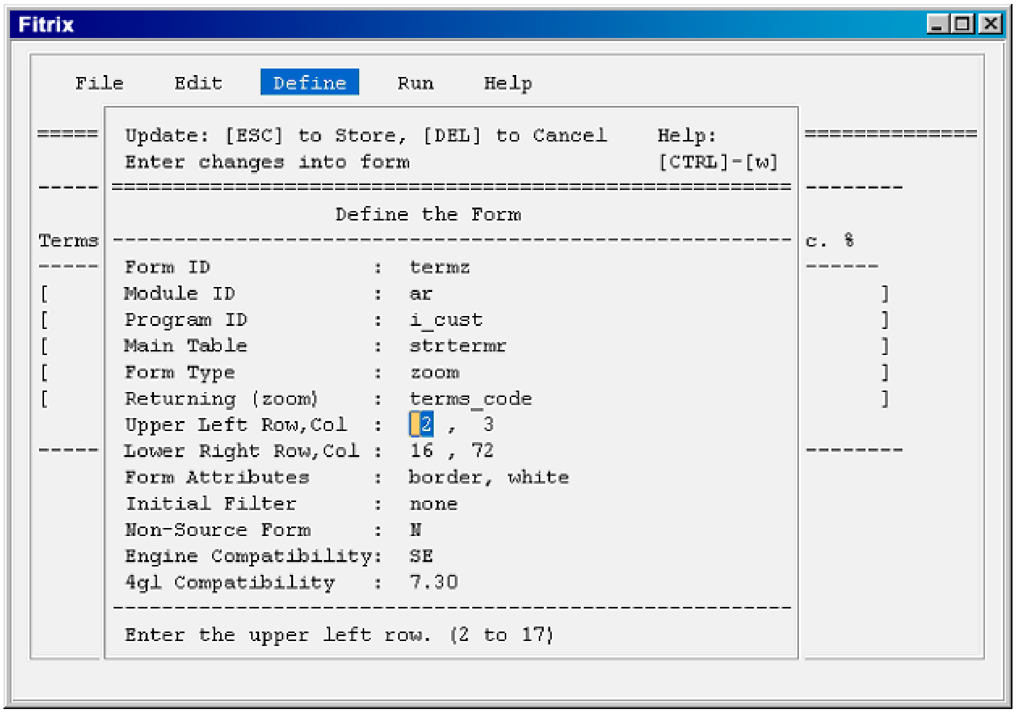
1. Select Form Defaults from the Define pull-down menu.
   The Form Defaults window appears.

2. Enter strtermr in the Main Table field.

   Zooms typically return values to the field from which they were invoked. Since you will be Zooming from the Terms Code field on you Customer Entry program, you must specify from which column the data will be supplied.

3. Add terms_code in the Returning (zoom) field.

   You can bypass the other fields on the window.



```
=====  Update: [ESC] to Store, [DEL] to Cancel    Help:       ==============
       Enter changes into form                    [CTRL]-[w]
-----  =======================================================  ---------
                     Define the Form
Terms  -------------------------------------------------------  c. %
-----   Form ID              :  termz                            ------
[       Module ID            :  ar                                     ]
[       Program ID           :  i_cust                                 ]
[       Main Table           :  strtermr                               ]
[       Form Type            :  zoom                                   ]
[       Returning (zoom)     :  terms_code                             ]
        Upper Left Row,Col   :  2 ,  3
-----   Lower Right Row,Col  :  16 , 72                          ---------
        Form Attributes      :  border, white
        Initial Filter       :  none
        Non-Source Form      :  N
        Engine Compatibility:   SE
        4gl Compatibility    :  7.30
       -------------------------------------------------------
        Enter the upper left row. (2 to 17)
```

4. Select Save Form from the File pull-down menu.

5. Select Generate 4GL from the Run pull-down menu.

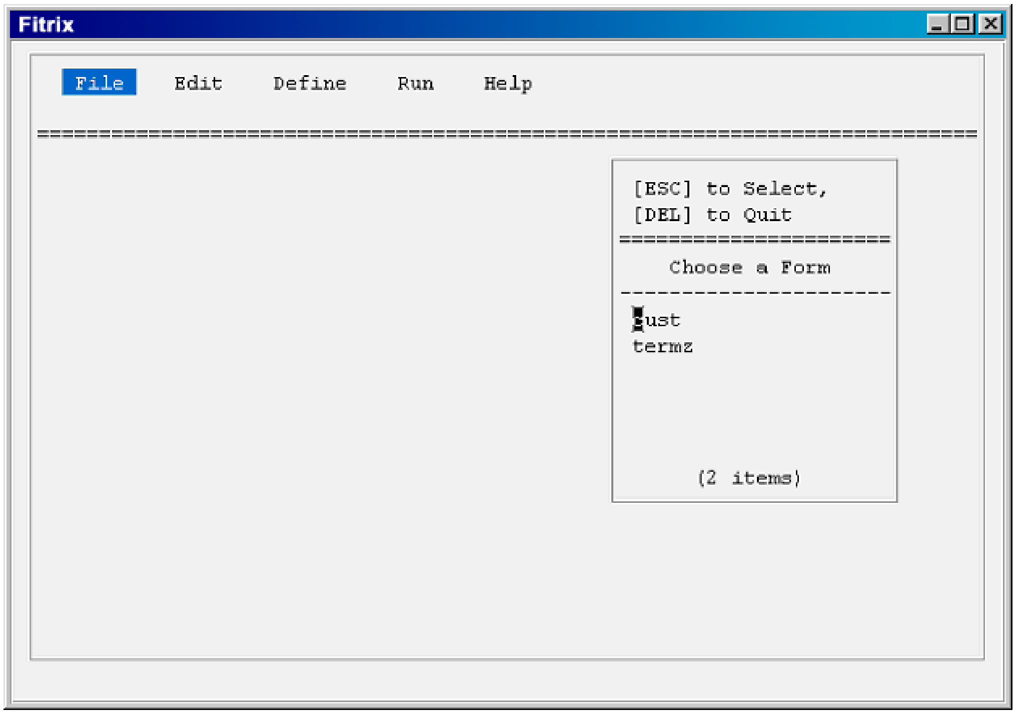
   The Generate 4GL: Enter Selection box appears.

6. Select termz.

Attach termz to the Terms Code Field

Now you must attach termz to the Terms Code field that you created on the Customer Entry program.

1. Select Open from the File pull-down menu and open the file that corresponds to your Customer Entry program (cust).



```
[ESC] to Select,
[DEL] to Quit
=======================
   Choose a Form
-----------------------
cust
termz



      (2 items)
```

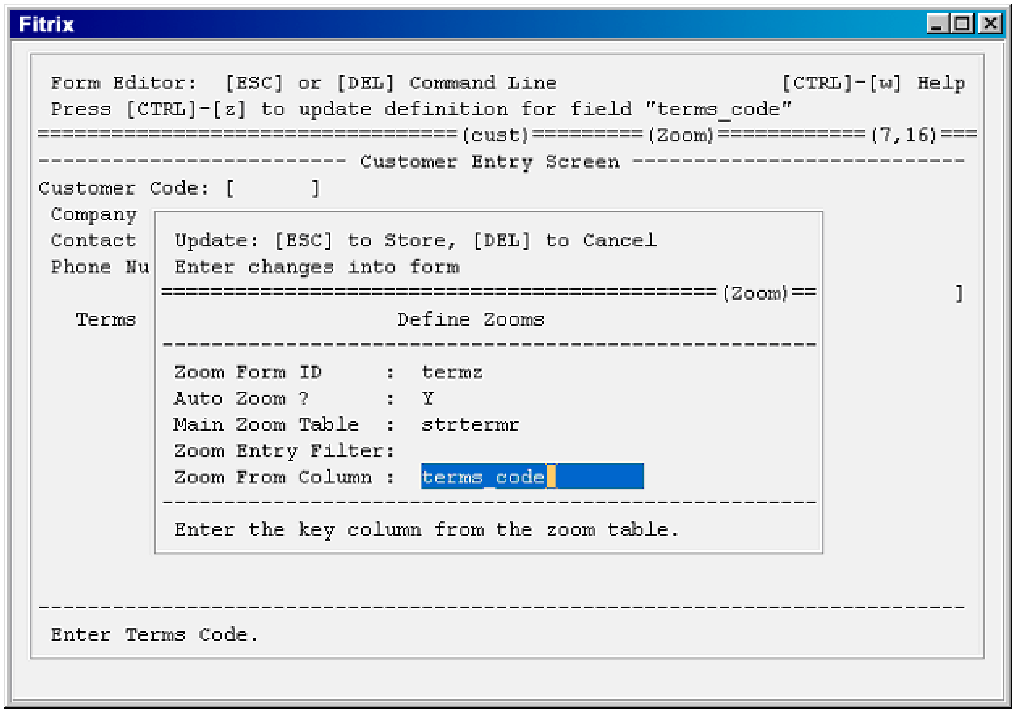
2. Place your cursor in the Terms Code field and press [CTRL]-[z].
   The Define Field pop-up menu appears.

3. Select Zoom… from the Define Field pop-up menu.
   The Define Zooms window appears.

4. Enter termz in the Zoom ID field.

5. Press [ENTER] in the AutoZoom ? field and enter strtermr in the Main Zoom Table field.

6. Specify terms_code in the Zoom From Column field and press [ESC] to save your zoom definition.



```
 Form Editor:  [ESC] or [DEL] Command Line                  [CTRL]-[w] Help
 Press [CTRL]-[z] to update definition for field "terms_code"
==================================(cust)=========(Zoom)============(7,16)===
------------------------- Customer Entry Screen ---------------------------
Customer Code: [       ]
 Company  ┌──────────────────────────────────────────────────────┐
 Contact  │ Update: [ESC] to Store, [DEL] to Cancel              │
 Phone Nu │ Enter changes into form                              │
          │============================================(Zoom)==│             ]
   Terms  │                    Define Zooms                      │
          │------------------------------------------------------│
          │ Zoom Form ID      :  termz                           │
          │ Auto Zoom ?       :  Y                               │
          │ Main Zoom Table   :  strtermr                        │
          │ Zoom Entry Filter:                                   │
          │ Zoom From Column  :  terms_code                      │
          │------------------------------------------------------│
          │ Enter the key column from the zoom table.            │
          └──────────────────────────────────────────────────────┘

---------------------------------------------------------------------------
 Enter Terms Code.
```

Save, Generate, and Compile

1. Use the Save Form option under the File pull-down menu.

2. Select Generate 4GL from the File pull-down menu.
   The Generate 4GL: Enter Selection box appears.

3. Select ALL Forms Only box appears.
   The Local Forms Only box appears.

4. Select YES.
   As the *Screen* Generator runs, it builds code for both your zoom screen and your Customer Entry screen. (Please select Option 1 if the message globals.org or any other .org exists, appears)

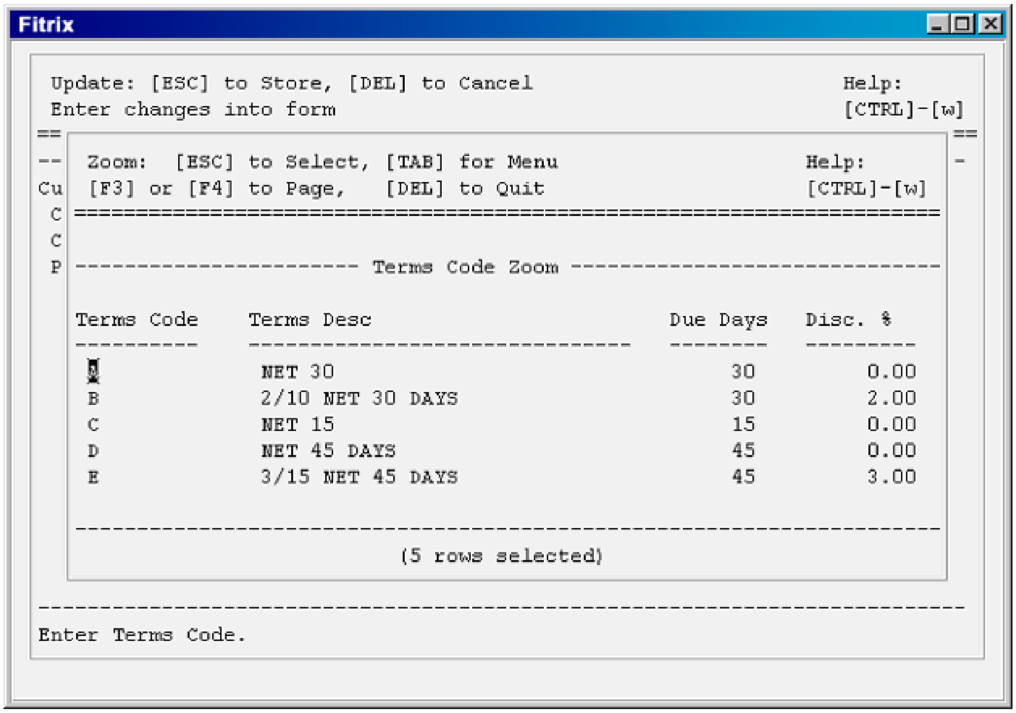
5. Select Compile 4GL from the Run pull-down menu.

Run Your Customer Entry Program

1. Select Run 4GL Program from the Run pull-down menu.
   The Customer Entry program starts.

2. Use Find to select an existing customer and select Update.

3. From the Terms Code field, press [CTRL]-[z] and press [ESC].
   The Terms Code Zoom appears.



```
 Update: [ESC] to Store, [DEL] to Cancel                          Help:
 Enter changes into form                                          [CTRL]-[w]
==                                                                           ==
--   Zoom:  [ESC] to Select, [TAB] for Menu                    Help:         -
Cu   [F3] or [F4] to Page,   [DEL] to Quit                     [CTRL]-[w]
 C  ==================================================================
 C
 P  ---------------------- Terms Code Zoom -----------------------------

    Terms Code    Terms Desc                     Due Days   Disc. %
    ----------    ------------------------------ --------   ---------
     A              NET 30                            30        0.00
     B              2/10 NET 30 DAYS                  30        2.00
     C              NET 15                            15        0.00
     D              NET 45 DAYS                       45        0.00
     E              3/15 NET 45 DAYS                  45        3.00

    ------------------------------------------------------------------
                              (5 rows selected)

-------------------------------------------------------------------------
Enter Terms Code.
```

4. Use the termz a few times. When finished, quit out of Customer Entry and the Form Painter.

Chapter 3

Creating Triggers

- Trigger Overview
- Understanding the Trigger Concept
- Creating Triggers
- Merging Triggers into Code

Trigger Overview

In most cases, you can use the Form Editor in the Form Painter to accomplish everything an input program requires. The Form Editor lets you:

- define input fields
- specify field attribute logic, such as whether the field can be entered
- attach zoom screens
- attach lookups to validate input values

On occasion, however, you must make custom enhancements to an input program that you cannot create in the Form Editor. For example, you might want to include some of the following enhancement types:

- after field logic
- before field logic
- after input logic
- after change in logic
- before input logic
- after row logic
- event handling logic

You can create all these enhancements using triggers, which are essentially code-level modifications to an input program.

Understanding the Trigger Concept

Triggers are enhancements made directly to the source code generator from the *Screen* Code Generator. A trigger is an automatic way of placing code-level enhancements into the source code.

Triggers are named for logical points in the code. The following list contains some common triggers:

- after_field
- before_field
- after_input
- before_input
- on_event

Triggers get placed in trigger (*.trg) files. A trigger file functions much like a form specification (*.per) file. Both contain instructions that the *Screen* Code Generator reads and understands.

A single trigger file can contain more than one trigger.

Triggers do not require you to be an expert on code structure. You simply work with the Form Painter to define the logical points at which your triggers act.

Trigger (*.trg) files should have the same name as the form specification file that they relate to. For example, the order.trg file relates to the order.per form specification file.

Creating Triggers

Creating triggers is a straightforward task. There are two ways you can construct triggers:

1. You can use the Form Painter.

2. You can create them manually in a trigger (*.trg) files.

Perhaps the best way to write your first trigger is with the Form Painter; it provides the simplest environment to learn about trigger creation.

Using the Form Painter to Create a Trigger

Before you create a trigger using the Form Painter, you should create a form image and form specification file.

Once you create a program from which to work, you can define a trigger.

To add a new trigger using the Form Painter:

1. Select Triggers >> from the Define pull-down menu.

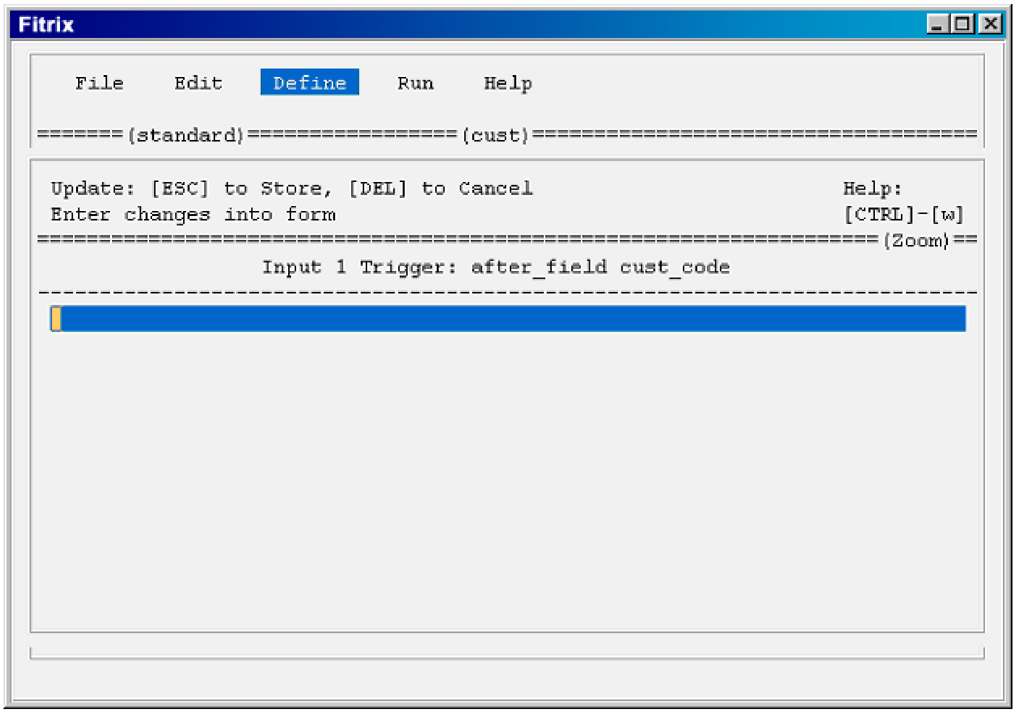
   If your screen type contains more than one input area, the Choose a Trigger Class pop-up menu appears.

2. Select the input area for your trigger.

   The Choose a Trigger pop-up menu appears.

3. Select the trigger you want to create.

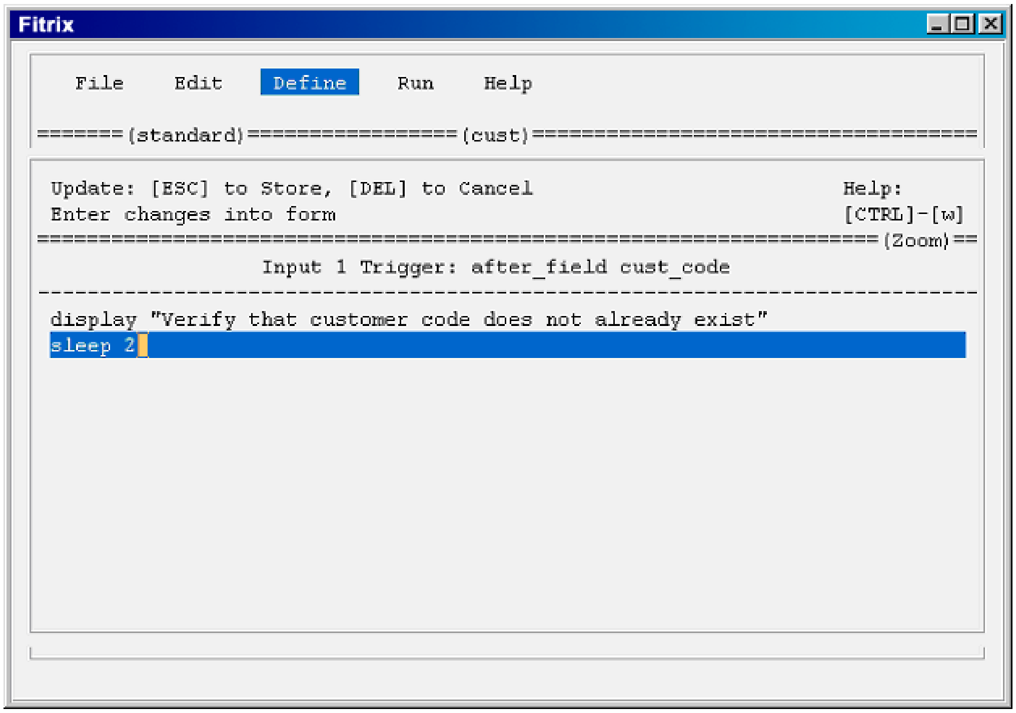
   Depending on the trigger you select, subsequent pop-up menus appear. For example, if you select the after_field trigger, the Choose a Field pop-up menu appears. After you choose a field, the Form Painter opens the Trigger Editor.



```
=======(standard)=================(cust)====================================
Update: [ESC] to Store, [DEL] to Cancel                              Help:
Enter changes into form                                              [CTRL]-[w]
==================================================================(Zoom)==
                 Input 1 Trigger: after_field cust_code
---------------------------------------------------------------------------
```

4. Enter the custom 4GL logic of your trigger using the Trigger Editor.

   For example, after the Customer Code field, you might want to display a message to check whether the Customer Code already exists. With the Trigger Editor, you can specify 4GL logic that displays this information. (This is just an example. The actual code will be dependent on the specific situation)



```
=======(standard)=================(cust)====================================
Update: [ESC] to Store, [DEL] to Cancel                           Help:
Enter changes into form                                           [CTRL]-[w]
====================================================================(Zoom)==
                Input 1 Trigger: after_field cust_code
----------------------------------------------------------------------------
display "Verify that customer code does not already exist"
sleep 2
```

5. Once you enter your trigger code, press [ESC] to store your trigger.

   Your trigger gets saved to a trigger (*.trg) file.

Creating Triggers by Hand

After a while, you might find it faster and more convenient to create triggers manually. That is to say, you might want to create trigger files directly using vi or some other UNIX text editor. Creating triggers by hand can be as simple as using the Form Painter as long as you follow the correct syntax. Always end each trigger with a ";" (semi-colon).

All triggers follow the same general syntax;

```
input #
        trigger argument
                custom 4gl logic …
                ;
```

Where # indicates the input area number, *trigger* indicates the trigger command, and *argument* indicates any argument that the trigger accepts.

For example, the following after_input trigger displays a short message:

```
input 1
        after_input
                display "after input logic"
                sleep 2
                ;
```

Some triggers accept arguments. For example, this trigger accepts a field name (company) as a trigger command argument:

```
input 1
        after_field company
                display "after field logic"
                sleep 2
                ;
```

Merging Triggers Into Code

Once you create a trigger, you can merge it into your source code. To merge a trigger, however, you do not need to regenerate all your code. You can simply run either the make utility (fg.make) or the Featurizer (fglpp).

If you are using the Form Painter, simply select the Compile 4GL option under the Run pull-down menu. If you are working from the command line, type:

```
fg.make
```

or:

```
fglpp
```

Both commands initiate the Featurizer. The Featurizer reads your trigger (*.trg) file and places your code enhancements into the generated source code. When you run fg.make, the final source code (*.4gl) files contain your enhancement logic. The Featurizer saves your original source code in files with and *.org extension.

Exercise 3

Objective: To add a simple before_input trigger to the Customer Entry program.

Open cust.per in the Form Painter

1. Go to $fg/accounting/ar.4gm/i_cust.4gc directory;
   cd $fg/accounting/ar.4gm/i_cust.4gc

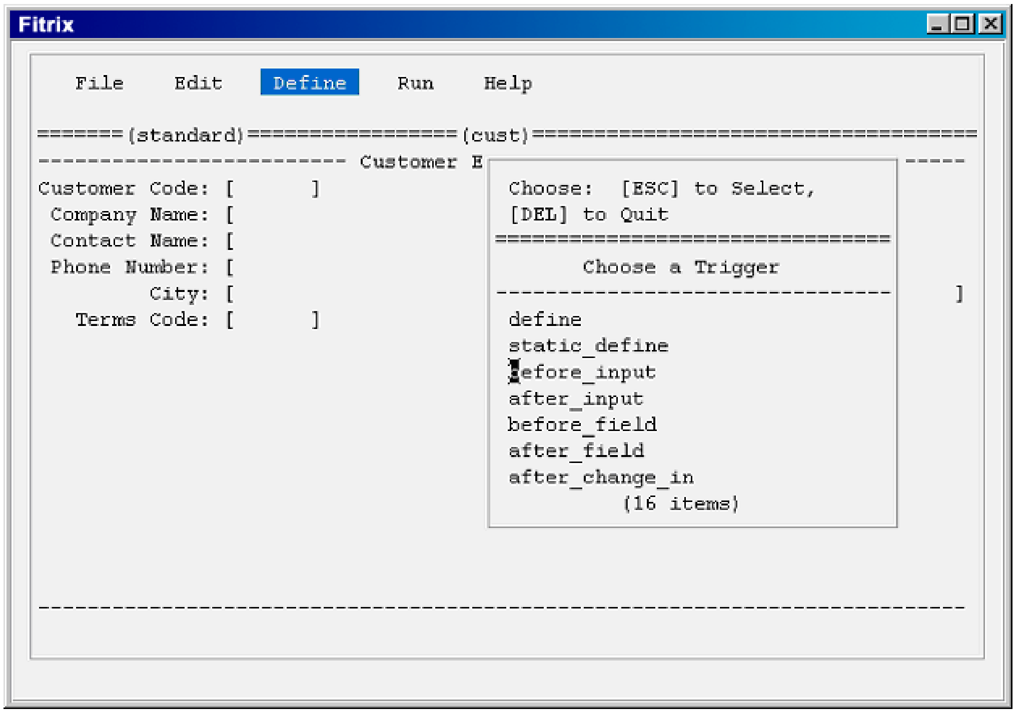
2. Start the Form Painter.

3. Select Open from the File pull-down menu to load cust.

Create a before_input Trigger

1. Select Triggers >> from the Define pull-down menu.
   The Choose a Trigger Class box appears.

2. Select Input Area 1 from the Choose a Trigger Class box.
   The Choose a Trigger list box appears.

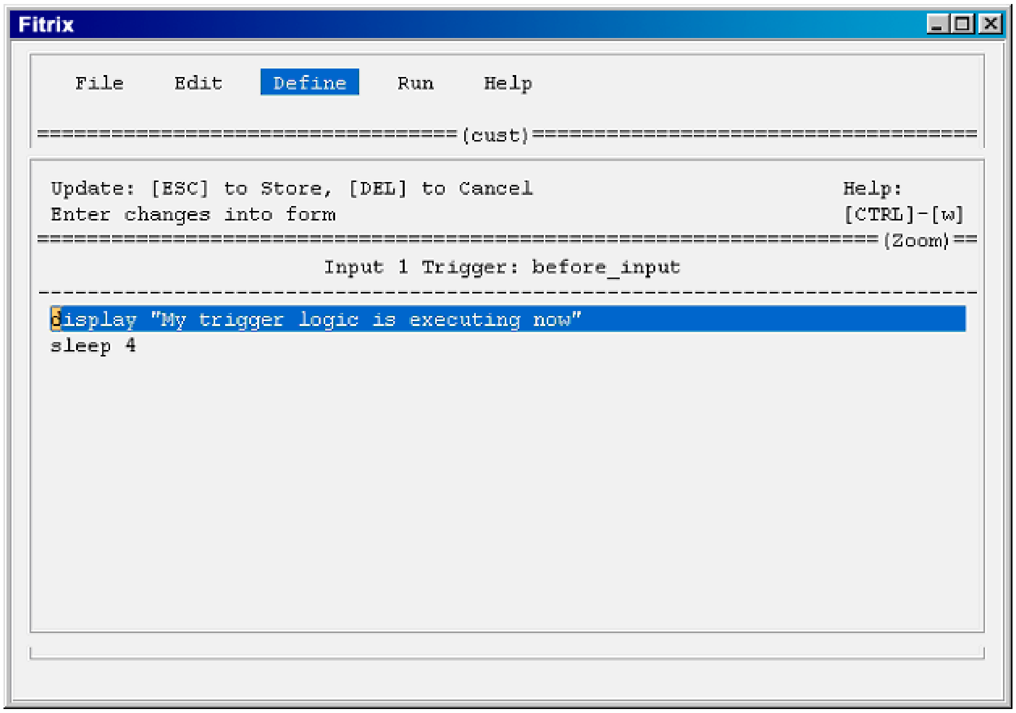


Fitrix

```
  File    Edit    Define    Run    Help

=======(standard)=================(cust)===================================
------------------------- Customer E┌─────────────────────────────────┐-----
Customer Code: [      ]            │ Choose:  [ESC] to Select,       │
 Company Name: [                   │ [DEL] to Quit                   │
 Contact Name: [                   │=================================│
 Phone Number: [                   │       Choose a Trigger          │
         City: [                   │---------------------------------│    ]
   Terms Code: [      ]            │ define                          │
                                   │ static_define                   │
                                   │ before_input                    │
                                   │ after_input                     │
                                   │ before_field                    │
                                   │ after_field                     │
                                   │ after_change_in                 │
                                   │          (16 items)             │
                                   └─────────────────────────────────┘

---------------------------------------------------------------------------
```

3. Select before_input from the Choose a trigger list box.
   An editing window appears.

4. Complete a “display” statement as follows:



```
=================================(cust)===================================
Update: [ESC] to Store, [DEL] to Cancel                          Help:
Enter changes into form                                          [CTRL]-[w]
=====================================================================(Zoom)==
                     Input 1 Trigger: before_input
----------------------------------------------------------------------------
display "My trigger logic is executing now"
sleep 4
```

5. Press [ESC] to save this before_input trigger.
   The Choose a Trigger list box appears again.

6. Press [DEL] to close the Choose a Trigger box.

7. Select Save Trg File from the File pull-down menu.

Compile the Code

- Select Compile 4GL from the Run pull-down menu.

The compilation utility calls the Featurizer (which is a code merging utility). The Featurizer merges your custom “display” logic into the generated source code.

Run the Customer Entry Program

- Select Run 4GL Program from the Run pull-down menu.

The Customer Entry program starts.

Note

When you make a change to a form (such as adding a field or field label), you must rebuild the program by running both the *Screen* Code Generator and fg.make. If you are only adding custom code via triggers, save the trigger file then run the fg.make. The *Screen* Code Generator is not required.

Check the before_input Trigger

1. Select Add from the ring menu.

   Your custom “display” logic appears at the bottom of the screen.

2. Finish adding the record.

3. Use Find to select a record and select update.

   Again, you custom logic appears.

4. Quit the program and the Form Painter.

Examine header.4gl

1. In the program directory that you created (i_cust.4gc), use vi to open header.4gl.

2. Search for before_input.

   Notice that your custom logic is inserted just before the input command:

   ```
   #_before_input
   # FGLPP BEGIN cust.trg (.4gc)
   display "My trigger logic is executing now"
   sleep 4
   # FGLPP END   before_input
   #_end

   #_input – Main input loop
   ```

   The #_characters mark a trigger tag. In other words, these symbols define locations where triggers can be inserted.

3. Using vi, search for other trigger tags.

   This step familiarizes you with the types of triggers that are available. You will be adding custom logic to some of these locations at a later time.

4. Exit header.4gl

Chapter 4

Compiling Generated Code

Compiling Generated Code

- Compiling generated code means turning 4GL source code and triggers into a runnable program. You have the ability to do this for a single program or a group of programs.

- You compile code using the Make Utility. This utility is run with the fg.make command. When you run fg.make, it completes the following tasks:

- Merges Custom Code: The fg.make command calls the Featurizer (fglpp) program. The Featurizer merges custom code into your program source code.

- Compiles Source Code and Form Specification Files: fg.make also compiles both your source code (*.4gl) files and form specification (*.per) files.

- Links Local Function Calls to Library Functions: The fg.make command resolves library function calls in locale (i.e., source code) to their corresponding library functions.

- Produces Runnable Program File: The last task of fg.make is to construct a runnable program file. The fg.make command creates a different program file depending on the type of development system you are using. In the case of Four J's BDL (4GL), a *.42r file is created. In the example, an i_cust.42r will be created.

The final three tasks are controlled by the standard UNIX make utility, which is called by fg.make. In general, the UNIX make utility tracks the dependencies that files have to each other.

The UNIX make utility uses a specification file of its own. This file, called the Makefile, contains all the instructions necessary for make to work. You do not have to create the Makefile; the Screen Code Generator creates it automatically.

For the most part, you do not need a complete understanding of the UNIX make utility in order to use it. You should simply realize that it is called from the fg.make command and it produces a program file that you can run.

The fg.make command uses a number of command flags:

```
fg.make [-h]  [-F | -R]  [-L library]  [-m ( n | o | f | of)]  [-f]  [-a]
```

-h Prints an entire list of fg.make command flags.

-F Forces fg.make to compile using the INFORMIX-4GL C Compiler.

-R Forces fg.make to compile using the INFORMIX-4GL Rapid Development System.

-L library Specifies the name of any additional libraries you want fg.make to link in.

-mn Does everything except merge code. In other words, when you use the –mn flag, the Featurizer is not called.

-mo Runs the Featurizer (merges code) but does not perform a compilation.

-mf Overrides timestamp comparison logic and forces a custom code merge.

-mfo Overrides timestamp comparison logic and forces custom code merge. This flag does not perform a compilation, however.

-f Performs a fast link. You should only use this flag in compiles where no new calls to library function have been added.

-a Causes all files to be compiled regardless of dependencies.

The Makefile

It does help, though, to have a working knowledge of the Makefile. The Makefile contains several sections. Each section supplies make with information about your program.

```
######################################################################
# Copyright (C)
# All rights reserved.
# Use, modification, duplication, and/or distribution of this
# software is limited by the software license agreement.
# Sccsid:  %Z%  %M%  %I%  Delta: %G%
######################################################################
# Screen Generator version: 4.12.UC1

#  Makefile for an Informix-4GL program

#_type - Makefile type
TYPE = program

#_name - program name
NAME = i_cust.4ge

#_objfiles - program files
OBJFILES =  globals.o header.o main.o midlevel.o termz.o

#_forms - perform files
FORMS =     cust.frm termz.frm

#_libfiles - library list
LIBFILES =  ../lib.a \
            $(fg)/lib/scr.a \
            $(fg)/lib/user_ctl.a \
            $(fg)/lib/standard.a

#_globals - globals file
GLOBAL =    globals.4gl

#----------------------------------------------------------------------

#_all_rule - program compile rule
all:
       @echo "make: Cannot use make.  Use fg.make -F for 4GL compile."
```

As you can see, the Make file lists the files necessary to create your program, e.g., the LIBFILES section shows all the libraries used by your program (lib.a, scr.a, user_ctl.a and standard.a).