

Chapter Thirteen

MESSAGES, BARS, PLOTS & SOUND

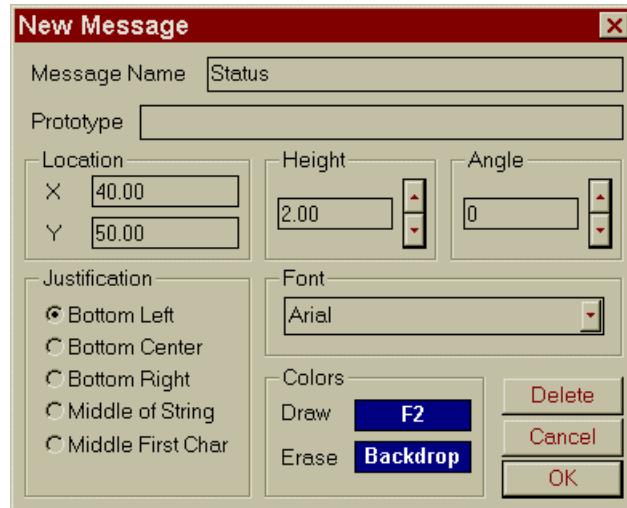
This chapter describes the **write** command, which changes text on the screen during an animation; the **set bar** command, which sets the height or width of a single bar in a bar graph; the **plot** command, which adds a line segment to an x-y graph; and the **play** command, which can play .wav files providing background music, narration, alarms or special effects

Adding and Editing Messages

A **Message** contains text that can be changed during the execution of an animation. Messages are often used to display statistics, e.g. “Average Time in System = 10.5 Minutes,” or equipment status, e.g., “Idle.” Messages can be defined as layout elements or as elements of a Proof Class. In the latter case, each Object created from the Class has its own unique Message text. This feature is often used to “tag” moving objects with identifying names. For example, in an animation of air traffic, it would be very nice to identify airplanes by airline and flight number. Messages can be helpful for debugging simulations and animations, as well.

Once you have defined a Message, you use the **write trace stream** command, described in the next section, to display or alter it during an animation.

To add a Message, first go to **Draw Mode** (or **Class Mode**, if the Message is to be part of a Class). Click the **M Add Message** button. You will be prompted to enter a name for the Message. You will use this name in **write** trace stream commands to supply text for the message during the course of an animation. After you have supplied a name for the Message, a dialog box that is similar to the **Text** toolbox will appear:

The image shows a 'New Message' dialog box with a red title bar and a close button. It contains several input fields and controls: 'Message Name' with the text 'Status', a 'Prototype' field, 'Location' with 'X' at 40.00 and 'Y' at 50.00, 'Height' at 2.00, 'Angle' at 0, 'Justification' with radio buttons for Bottom Left (selected), Bottom Center, Bottom Right, Middle of String, and Middle First Char, 'Font' set to Arial, 'Colors' with 'Draw' set to F2 and 'Erase' set to Backdrop, and buttons for Delete, Cancel, and OK.

There are two differences between the Message dialog box and the Text dialog box. The Message dialog box has controls near its top for specifying a **Message Name** and a **Prototype**. The Text dialog box has only a Text control near its top. In the Text dialog box, the Text control specifies static text that is a permanent part of the current layout. In the Message dialog box, the Prototype control allows you to view (in Draw Mode or Class Mode only) “typical” text at a specified position in the layout. For example, you might specify “12.345” as the prototype for a Message that is used to display numeric values that you plan to write using two digits to the left of the decimal point and three to the right, just so you can see in Draw Mode how it will look. The text displayed when you run the animation will differ from the prototype. In Run or Debug mode, nothing is displayed for a given Message until the first **write** command for the Message is processed.


The second difference between the Text and Message dialog boxes is that a Message has two colors, while Text has only one. A Message’s **Erase** color is used when the content of the Message is changed by a **write** command. Before the new text is displayed (using the Draw color), the old text (if any) is rewritten in the Erase color. The default Erase color is Backdrop, because most Messages are placed over Backdrop-colored areas.

If you have a Message that lies atop a filled region, you will need to set the Message's Erase color to match that of the filled region "underneath" the Message.

When you create a new Message, after you have specified the Message's name (in response to a prompt from Proof to do so), a red-colored text position marker will begin following the mouse around the screen. When you are satisfied with the Message's position, click the mouse to "nail down" the Message's position.

You should not position a Message in a place where it could get "run over" by Objects or interfered with by Bars or other Messages.

You may rename the active Message by simply typing a new name in the **Message Name** control. As always, you must save your layout to make the change permanent.

Click the  **Add Message** button, or click in an open area of the screen when you are satisfied with the current Message and would like to add another Message.

To edit an existing Message, click on the prototype text string of the Message that you would like to edit. Now you can change any of the items in the dialog box.

Altering a Message's Contents During an Animation

The **write** trace stream command is used for altering the contents of a Message during an animation. The syntax of the **write** command is

write *messagename textstring...*

write *messagename(ObjectID) textstring...*

The second form is used only for Messages that are defined within a Class. For such Messages, you must specify the individual Object for which the Class's Message is to be altered.

Here are some examples of the **write** command:

```
write STATUS BREAKDOWN
```

```
write Util22 0.698
```

```
write Warning Channel Capacity Exhausted
```

```
CREATE widget 1
```


```
write Status(1) Active
```

The *messagename* must match that of a Message defined in the active layout file in order to use **write**.

The *textstring* can include blanks. The *textstring* always begins with the second character after the message name (the first being a blank space), and ends with the last non-blank character in the trace stream line. If the *textstring* is omitted, the current content of the Message (if any) is erased from the screen.

When you use **write** to display tabular statistics, remember that digits (0-9) are drawn with uniform widths in almost all fonts. However, we have seen fonts in which the digits are of non-uniform width. Such fonts are unacceptable for columnar output.

Exercise 13-1: Writing a Message in an Animation

Let's revisit the **mystars** model from Exercise 8-2 to experiment with the Message element and the **write** command. First, you'll need to modify **shapes.lay** (located in the **exercise** folder) to contain a Message. Bring up this layout, enter **Draw Mode**, and create a **Message** by clicking on the  button. **Name** the message "Status", and for the Prototype, type "Message goes here."

Position the message somewhere near the middle of the screen. **Save** your layout as **stars.lay**.

Now **Edit** your commands in the model file. Add a statement to your model or program that causes this line to be placed into the trace stream every time a star is turned on:

```
write Status STAR 1 ON
```

Make a similar change to insertion of the line

```
write Status STAR 1 OFF
```

each time a star is turned off.

Now rerun your model or program. Once the simulation has completed executing, bring up `stars.lay` & `stars.atf`, and run the animation to see your Message in action.

About Bars

A **Bar** is typically used to graphically represent a changing quantity. With one or more Bars in your layout and the appropriate commands in your trace stream, you can develop a pleasing “dancing bar graphs” animated display.


A Bar is displayed as an orthogonally-oriented, filled rectangle. Bars will rotate if the View Rotation is changed, but if rotated they will not appear filled, even if all you do is switch to an isometric viewing perspective. The base orientation of a Bar cannot be rotated with Box Edit or from the trace stream.

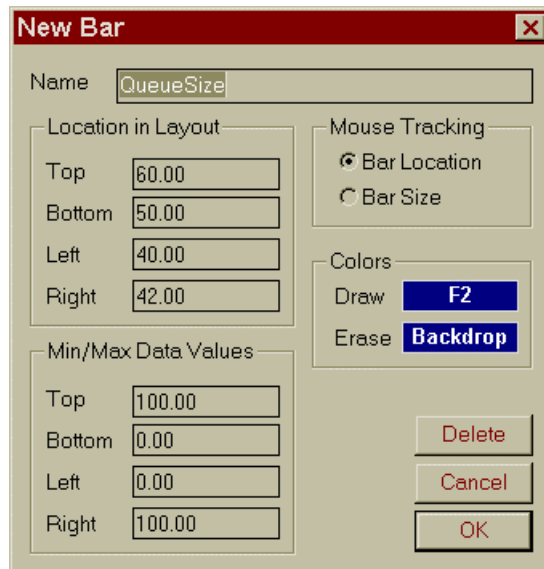
A Bar is not an Object. It can be modified only through the `set bar` command. Bars cannot be included in Classes. Accordingly, the Class Mode toolbar has no “Bar” button.

Bars are measured in two ways. First, the physical location of a Bar (top, bottom, left, and right positions) is specified in animation coordinates. Second, the top, bottom, left, and right data values are specified in units of your choice. If you are using a Bar to depict the utilization of a server, you might choose top and bottom values of 100.0 and 0.0 or 1.0 and 0.0 for ease of manipulation. Because this usage is so commonplace, the defaults are bottom=0.0, top=100.0, left=0.0, and right=100.0. You can choose any values you want, except that the top value must always be greater than the bottom value, and the right value must always be greater than the left value.

The reason Bars are measured in two ways is to maintain independence of the Bar’s physical position and the data it represents. If only a Bar’s physical position could be specified, every time you edited the position or size of a Bar (in Draw Mode), you’d have to change the computations that mapped your data values into physical screen positions. When two measures are used, once you’ve defined the extreme of the data to be depicted, you can move or resize or move a Bar without changing your model logic.

Adding and Editing Bars

You use the  Add Bar button to add a Bar in Draw Mode. Proof will then prompt you to specify a name for the Bar. The name is needed for use in `set bar` trace stream commands. After you have specified a name for the Bar, a dialog box similar to the following will appear:

The image shows a 'New Bar' dialog box with a red title bar and a close button. It contains several sections: 'Name' with a text field containing 'QueueSize'; 'Location in Layout' with four input fields for Top (60.00), Bottom (50.00), Left (40.00), and Right (42.00); 'Min/Max Data Values' with four input fields for Top (100.00), Bottom (0.00), Left (0.00), and Right (100.00); 'Mouse Tracking' with two radio buttons, 'Bar Location' (selected) and 'Bar Size'; 'Colors' with 'Draw' and 'Erase' buttons, where 'Draw' is labeled 'F2' and 'Erase' is labeled 'Backdrop'; and three buttons at the bottom: 'Delete', 'Cancel', and 'OK'.

When the dialog box appears, a rectangle with red handles at each corner will appear on the screen. The rectangle is in the current pen color. This rectangle is a visible representation of the maximum size of your Bar.


You will notice that the lower left corner of the Bar tracks the cursor as you move the mouse around. The corner will snap to the nearest grid point, if applicable. When you click the mouse, the Bar will stop moving. You can move the Bar again by clicking and holding the mouse near any corner, and dragging that corner to a new location.

In order to change the maximum size of the Bar, click on **Bar Size** in the **Mouse Tracking** box. When Bar Size tracking is enabled, you can drag the Bar by any of its corners, and the diagonally opposite corner will remain fixed. Mouse tracking is subject to the current Snap-to-Grid setting.

Once you are satisfied with the location and shape of a Bar, you can specify the numerical equivalents for the minimum and maximum horizontal and vertical *values* of the Bar.

Like Messages, Bars have two colors, a Draw color and an Erase color. Each time a Bar is redrawn at a new size, it is first redrawn at its current size, in its Erase color. To change the color of the Bar, simply click on the **Draw** color button and select the desired color from the color palette pop-up menu that appears. Likewise, to change a Bar's Erase color, click on the **Erase** color button.

If you would like to rename the currently active Bar, simply type a new name into the **Name** control of the Bar dialog box.

Once you are satisfied with a Bar’s position, size, and values, you can move on to the creation of another Bar or to some other activity. If you’d like to add another Bar, click in an open area, or click on the  **Add Bar** button. Like all elements, Bars are subject to further editing, and also will not be saved permanently until you save the layout. To edit an existing Bar, simply click on it. The Bar will be selected and can be changed or moved in the same manner as a new Bar.

The Set Bar Command

The trace stream command for controlling a Bar’s “fullness” and/or its color is **set bar**. The syntax is

```
set bar barname level [ rate rate ] [ color c ] [ BGcolor c ]
```

```
set bar barname [ { direction level [ rate rate ] } ]* [ color c ] [ BGcolor c ]
```

```
set bar barname [ { direction rate rate } ]* [ color c ] [ BGcolor c ]
```

The asterisks (*) in the second and third forms shown above indicate that the elements enclosed in the preceding brackets may be given 0...4 times. *direction* can be *top*, *bottom left*, or *right*. In the first form shown above, no *direction* is specified, and *top* is assumed. In the second form, both a *level* and a *rate* are specified. In the third form, no *level* is specified, and the *rate* is applied starting at the current level.

A *level* and/or *rate* can be set on any of the Bar’s four edges. Thus, Bars can grow and shrink vertically, horizontally, or in any combination thereof. For convenience, you can specify Bar levels or rates for more than one direction in a single **set bar** command. If a rate is specified, it is interpreted as data units per unit of animation time. For example, if you’re using a time unit of seconds, and the data depicted on a given Bar ranges from zero to 100, a rate of 1.0 implies a one percent per second increase in the specified direction. For an example of the use of Bar rates, see the **barrates** animation in the **sample** folder.

Instead of, or in addition to level/rate specifications, you can specify the keyword **color** followed by a Proof Animation color name or number. This lets you change the color of a Bar during an animation.

The *level* determines the new shape of the bar based on interpolation between the minimum and maximum values as specified in Draw Mode. For example, if you specify a *top level*, then the height of the bar will be some fraction of the maximum height of the bar as originally drawn.

The fraction will be:

$$(\textit{level} - \text{bottom value}) / (\text{top value} - \text{bottom value})$$

A *level* greater than the top value (in this example) would be displayed as if it were equal to the top value.

Here are some examples of **set bar**.

set bar inventory 40

set bar inventory top 40

set bar inventory color L1

set bar temperature right 96

set bar temperature top 100 right 96 color F1

set bar TankLevel top 50 rate 3

set bar ShrinkingFast left rate 4 right rate -4

The initial values of **bottom** and **left** are the original bottom level and left level, respectively. (If you did not specify them in Draw Mode, these values are 0 by default.)

The initial values of **top** and **right** are equal to the initial values of **bottom** and **left**, respectively. So, a given bar is invisible until an appropriate **set bar** is processed.

When Proof Animation processes a **set bar** command for the first time for a given bar, all four values are used to initialize the Bar so that it reaches the maximum extent in all four directions. The extent(s) of the specified direction(s) is (are) then immediately modified according to the **set bar** command. If the level decreases, the difference in size between the current level of the Bar and the previous level are filled in using the **Erase** color.

Using Low-Priority Colors to Achieve Special Effects

By defining a Bar's color and erase color in the Layout as low-priority colors and partially overlaying the Bar with shapes drawn in higher priority colors, you can achieve special effects such as non-rectangular level indicators, because the Bar's low-priority colors will be masked by the high-priority colors. Take a look at the **tanks** animation in the **sample** folder. This animation superimposes a circular shape over a Bar to animate an end view of a circular tank.

Exercise 13-2: Creating an Array of Bars

The files `bars.lay` and `bars.atf`, in the `exercise` folder, animate a classic one-line, one-server queueing model. The animation displays simple statistical information. In the underlying model, the interarrival times are exponentially distributed with a mean of 16 minutes. This will cause about 30 arrivals per eight-hour day. We've run the model for 200 arrivals.

The service time is uniformly distributed between 12 and 18 minutes. The only delays in the system are the wait for the server and the service time.

To see Bars in operation, run the `bars` animation.

In `bars.atf`, the only commands are `time` and `set bar`. There are two Bars. The utilization Bar is called `UTIL`, and the queue contents Bar is called `QUEUE`. Each Bar is updated every five simulated minutes.


Notice that the animation displays the *moving average* of the utilization over a 50 minute interval. For a converging statistic such as utilization, the moving average approach provides more information for the person viewing the animation.

There is also a trace stream from a modified version of the model. In the modified model, the average interarrival time is 2.75 minutes, and there are six servers. The resulting trace stream is in `sixbars.atf`. This trace stream contains information to display one utilization Bar for each individual server in addition to the two Bars for utilization and overall queue contents.

Your job is to modify the layout to add six vertically-oriented Bars named `UTIL1`, `UTIL2`, `UTIL3`, `UTIL4`, `UTIL5`, and `UTIL6`. Use the default values (0 to 100) for the maximum extent of each Bar.

You can go directly into Draw Mode while running `bars`. As soon as you are in Draw Mode, save the layout using the name `sixbars.lay`. (This will prevent you from accidentally overwriting `bars.lay` when you finish your modifications.)

Notice that the two existing Bars are displayed at their full extent in Draw Mode. Bars are always displayed that way in Draw Mode.

Click the  **Add Bar** button and add a vertically-oriented Bar named `UTIL1`. You may wish to first reposition the existing text and/or Bars to make room for the six new Bars.

Now add the other five Bars. Hint: use **Box Edit**. Draw a box around `UTIL1` and click on Box Edit's **Copy** button. **Paste** once, then drag `UTIL2` to the desired location, then click on **Repeat** four times.

Pasting a named element (a Bar, a Message, a Plot, or a layout Object) causes a numeric suffix in the name to be created or incremented, as applicable. This automatic naming process always generates an Alert for each name. You can use the <Esc> key to make the Alerts go away more quickly.

Save the layout, then re-open and execute `sixbars.lay` & `sixbars.atf`.

About Plots


Plot layout elements define the framework for line plots or graphs in your layout. You can set the limits of the X- and Y-axes, plotting and axis colors, axis labeling, etc. in Draw Mode. The actual data line segments comprising a plot are created in a trace stream. Using the `plot` trace stream command, you can add new segments to a Plot or redraw existing segments.

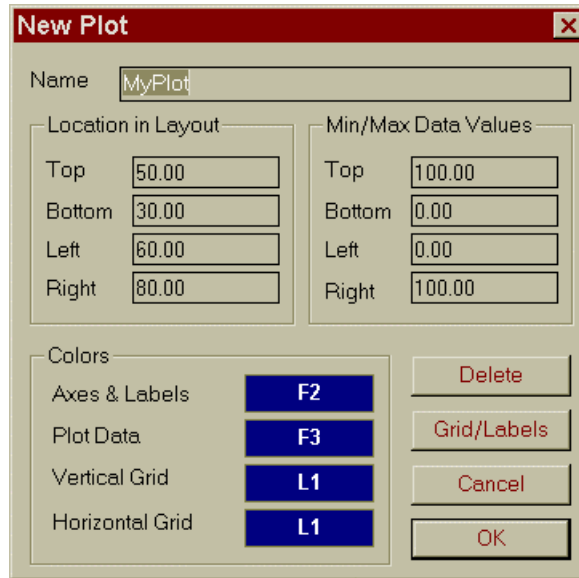
A Plot is not an Object. It can be modified only through the `plot` trace stream command. Plots cannot be added to Classes. Accordingly, the Class Mode toolbar has no **Add Plot** button.

Plots are measured in two ways. First, the physical location of a Plot (top, bottom, left, and right positions) is specified in animation coordinates. Second, the top, bottom, left, and right data values are specified in units of your choice. For example, if you are using a Plot to depict the contents of a queue over time, you should choose a top data value that will reasonably accommodate expected queue contents. Likewise, you should choose a right data value that will accommodate the expected duration of the animation. Portions of a Plot that fall outside the declared range of data values are ignored. Such occurrences are not treated as errors. You can choose any values you want, except that the top value must always be greater than the bottom value, and the right value must always be greater than the left value.

The reason Plots are measured in two ways to maintain independence of the Plot's physical position and the data it represents. If only a Plot's physical position could be specified, every time you edited the position or size of a Plot (in Draw Mode), you'd have to change the computations that mapped your data values into physical screen positions. When two measures are used, once you've defined the extreme of the data to be depicted, you can move or resize a Plot without changing your model logic.

Adding and Editing Plots

You use the  **Add Plot** button to add a Plot to your layout. When you do so, Proof will prompt you to specify a name for the Plot. The name is needed for use in `plot` trace stream commands. After you have specified a name for the Plot, a dialog box similar to the following will appear:



The 'New Plot' dialog box is shown with a red title bar and a close button. It contains several sections: 'Name' with a text field 'MyPlot'; 'Location in Layout' with four numeric input fields (Top: 50.00, Bottom: 30.00, Left: 60.00, Right: 80.00); 'Min/Max Data Values' with four numeric input fields (Top: 100.00, Bottom: 0.00, Left: 0.00, Right: 100.00); and 'Colors' with four buttons (F2, F3, L1, L1) and four more buttons (Delete, Grid/Labels, Cancel, OK).

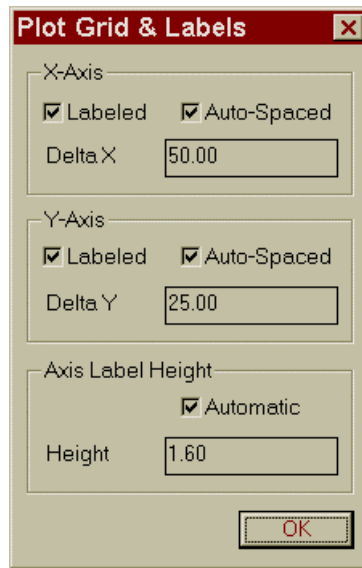
When the New Plot dialog box appears, a small plot will also appear on the screen. As you move the mouse, the Plot will follow the mouse, and you will be able to see the Top, Bottom, Left, and Right values change in the Location in Layout box within the Plot dialog box. Once you have positioned the Plot in the desired location, you can click the mouse to lock it in place. If you would like to move the Plot again, click and hold on the red dot at the intersection of the X- and Y-axes, and then drag the Plot to the new location.

By default, the X- and Y-axes (Left & Right and Bottom & Top, respectively in the Min/Max Data Values box) go from 0 to 100. To change any of these values, simply type a number into the appropriate control.

You can specify the colors used for plotting data, for erasing data, for drawing axes and labels, and for the vertical and horizontal grids by clicking on their respective buttons in the Color box. If you wish to suppress any of these features, simply specify its color as Backdrop.

If you wish to rename a Plot, simply type a new name into the Plot dialog box's Name control at any time.

To specify the characteristics of a Plot's Grid (not related to the global Proof Grid) or Labels, click on the **Grid/Labels** button, and a dialog box similar to the following will appear:



Axes can be labeled (can have a number at each grid interval) or unlabeled, and you can either let Proof automatically determine interval spacing, or specify the spacing yourself. (If your plot needs text labels or annotations such as a plot title or axis descriptions, you would add these separately using Text.)

Terminating Plot Specification

Once you are satisfied with your Plot, you can click the **OK** button in the Plot dialog box. Alternatively, if you want to define another Plot immediately, click the mouse in an open area of your layout, away from any layout element. You will then be prompted for a new Plot name, and the definition process will begin for the next Plot.

Editing a Plot

To edit an existing Plot, click on the top of its Y-axis, the right end of its X-axis, or the intersection of the X- and Y-axes. You can then edit the Plot exactly as you would edit a new plot. Remember, any changes you make will not become permanent until you save the current layout.

The Plot Trace Stream Command

The plot trace stream command is used to write the data line segments in a Plot. The syntax of the plot command is as follows:

plot *plotname* [#*segmentID*] *x1 y1 x2 y2* [**color** *c*]

plot *plotname* **clear**

The optional *segmentID* tag is used when individual data line segments will be redrawn (replaced) over time in the animation. For example, you could animate an Electrocardiogram (EKG) by plotting a number of small segments left-to-right and recycling through the segments when the EKG trace “wraps around” from the right edge of the plot to the left. When a *segmentID* is given, the previously plotted segment tagged with the same ID, if any, is erased by being rewritten in the Backdrop color before the new line is plotted.

x1 and *y1* are the coordinates of the starting point of the segment, and *x2* and *y2* are the ending points of the segment.

If you would like the segment to be drawn in a color that is different from the specified data color, an optional color can be specified in the plot command.

The **clear** option is used to completely erase all of the current data line segments in the given Plot. The erase is done by rewriting all of the lines in the **Erase** color.

Here are some examples of the plot command.

```
plot numqueue 50 26.6 60 57.4
```

```
plot Utilization 34.4 50.65.3 75.5 color F5
```

```
plot GRAPH1 #4 100 15 215 27 color F4
```


```
plot Mygraph Clear
```

In the third example, data line segment #4 of the Plot named GRAPH1 will be drawn or redrawn in F4 (Yellow).


In the last example, all of the data line segments are rewritten in the data background color, and numbered line segments (if any) are deleted from Mygraph.

Exercise 13-3: Creating a Plot

Part A

Run the **return** animation, located in the **exercise** folder. This animation depicts a single clerk processing customer returns. Your task is to add a Plot that shows the current number of customers in line each time a customer enters or exits the line. The customer currently being served is not considered “in line.” To set up the Plot, enter **Draw Mode**, and click the  **Add Plot** button. When you are prompted to enter a Plot name, type in “queue” (in lower case). Using the mouse, position the Plot in the empty area to the right of the returns desk. Click the mouse to place the Plot. Both the X- and Y-axes should be labeled and the **Plot Data** color should be F4. These can be set in the Plot toolbox. In **Min/Max Data Values** option box, specify a **Top** value of 10.0 (maximum expected queue contents), and specify a **Right** value of 360.0 (run length).

Using the mouse, **Drag** the right end of the X-axis until the X-axis shows 5 or 6 tick marks, and release the mouse button. If necessary, use the **Scrollbars** to locate the right end. Do the same for the Y-axis by dragging it by its top.

If you wish to add additional annotations, you can do so by clicking the  **Add Text** button. It would be nice to place the annotation “Time (Minutes)” below the X-axis and to place the annotation “Current Number in Queue” along the Y-axis. When you’re finished, your layout should look similar to Figure 13-1:

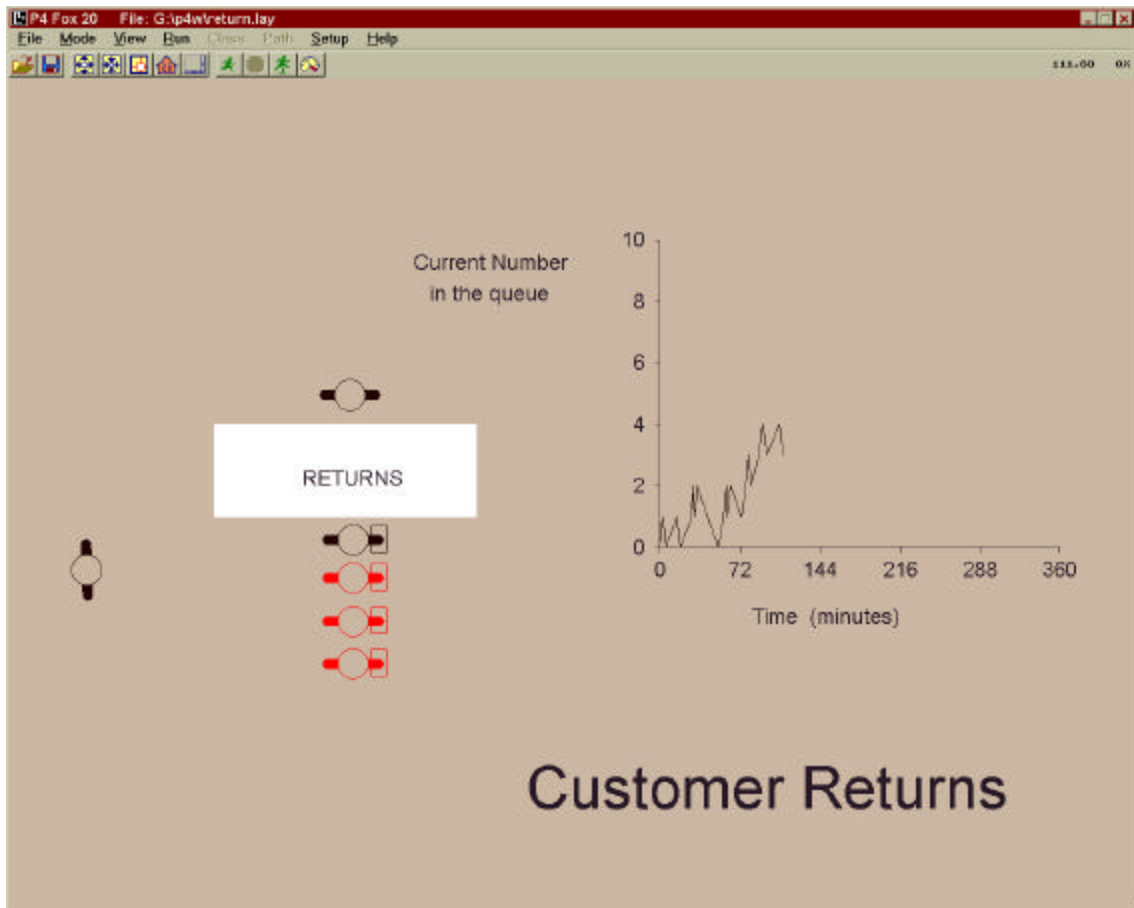


Figure 13-1.

Save your layout, and choose **Mode, Run** to enter Run Mode. Choose **File, Open Trace**, and select or type `return.atf`. Start the animation and watch your Plot in action

Part B

In Part B of this exercise, the Layout and Trace files are provided for you in the **sample** folder. This variation has two clerks at the returns desk. At time 0, the graph of the single clerk queue from part A is displayed on the Plot in color F4. As the animation runs, a second graph representing the number in queue of the two clerk system is overlaid on the same Plot in color F5. Run the `return2` animation to see how this works.

Adding Sound to an Animation

The `play` trace stream command can be used to play `.wav` sound files during an animation. The syntax of the `play` command is as follows:

play [**asynchronous**] *wavefile*[.wav]

play off

The `asynchronous` option is used when you want the playing of sound to overlap ongoing animation. This is almost always the case during an animation, so you will almost always use the `asynchronous` option in a trace stream. However, note that there is also a `play` command for use in presentations (see Chapter 17). In a presentation, sound files are almost always played synchronously; that is, the presentation pauses until the sound file plays to completion. Since sound files are used more commonly in presentations than in trace streams, we made synchronous playing the default

If you wish to terminate the playing of the current sound file, you can use `play off`.

Exercise 13-4: Adding Sound to an Animation

There's a file named `tada.wav` in the `sample` folder. Select any of the animations you have developed so far and add the command `play asynchronous tada` to the start of the animation's trace file. You might need to copy this file to the `exercise` folder, depending on which animation you choose. Run the animation and see (hear) what it does

Older Forms of Sound

Early (non-Windows) releases of Proof supported a `sound` command, which allowed you to control a PC's tone generator. The `sound` command allowed you to specify the pitch of the note to be played. Pitch was specified either in Hertz, e.g., "`sound 440`", or as a note name, e.g., "`sound A#4`" (A-sharp in the fourth octave of a piano).

Windows does not provide for end-user control of a PC's tone generator. Accordingly, there is no support of the `sound` command in Proof Release 4.0.