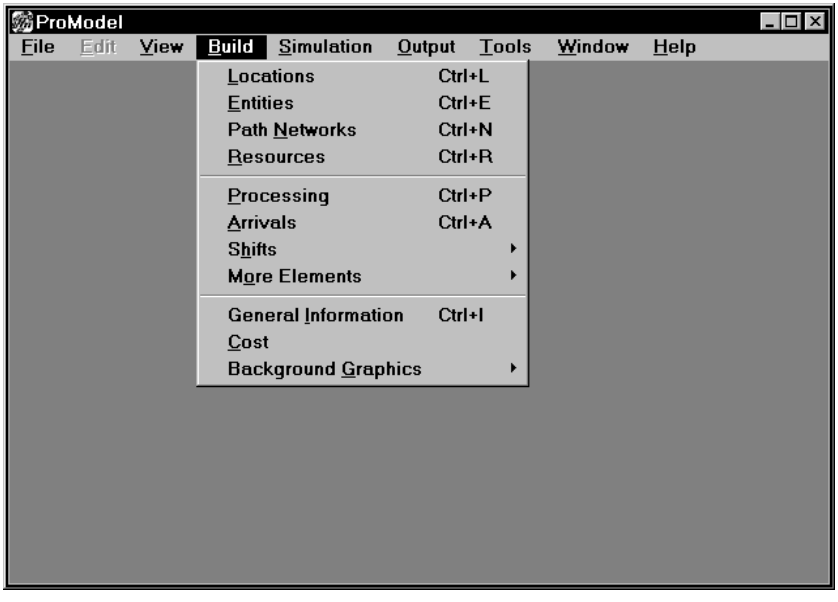
Chapter 7 Defining General Elements

CHAPTER CONTENTS

CHAPTER CONTENTS

7.0.1 Build Menu

The Build Menu is the gateway to all modeling elements used to define a model. Through this menu you specify the locations, entity types, arrival rates, resources, path networks, downtimes, processing logic, and other elements such as variables, attributes, and arrays that provide the flexibility needed to model your system.



How To **Access the Build Menu:**

- Select **Build** from the menu options bar.

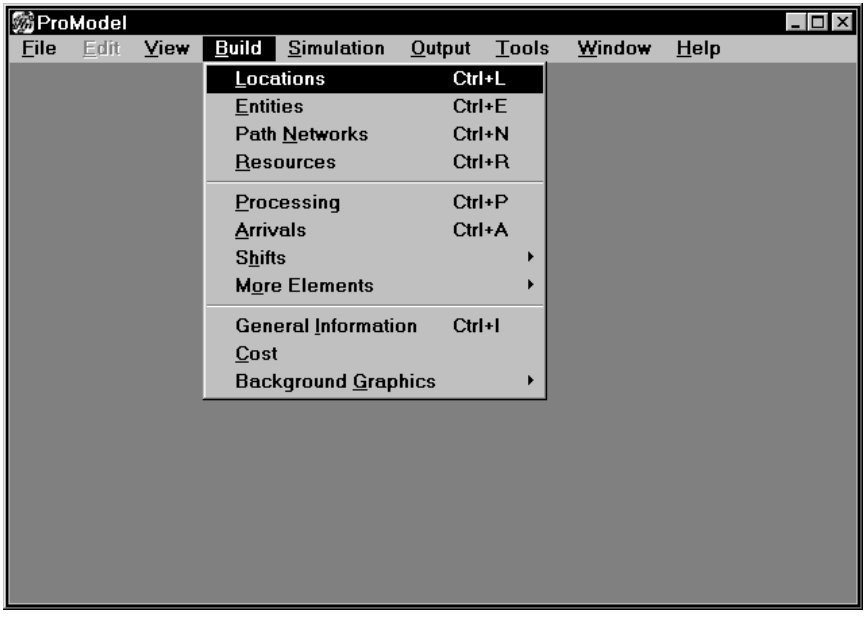
Each selection from the Build Menu is covered in detail in the following sections of this chapter.

7.1 Locations

Locations represent fixed places in the system where entities are routed for processing, storage, or some other activity or decision making. Locations should be used to model elements such as machines, waiting areas, work stations, queues, and conveyors.

Every location has a name and a name-index number. The name-index number is the location's numerical position in the list of locations. Logic which refers to a location, such as routing logic, can use either the location's name, or the LOC() function to refer to the location. The LOC() function allows a location whose index number has been stored in an attribute or variable to be referenced. See *Loc()* on page 180 of the *ProModel Reference Guide* for more information.

Locations are defined in the Locations Editor, which is accessed through the Build menu.



How To **Create and edit locations:**

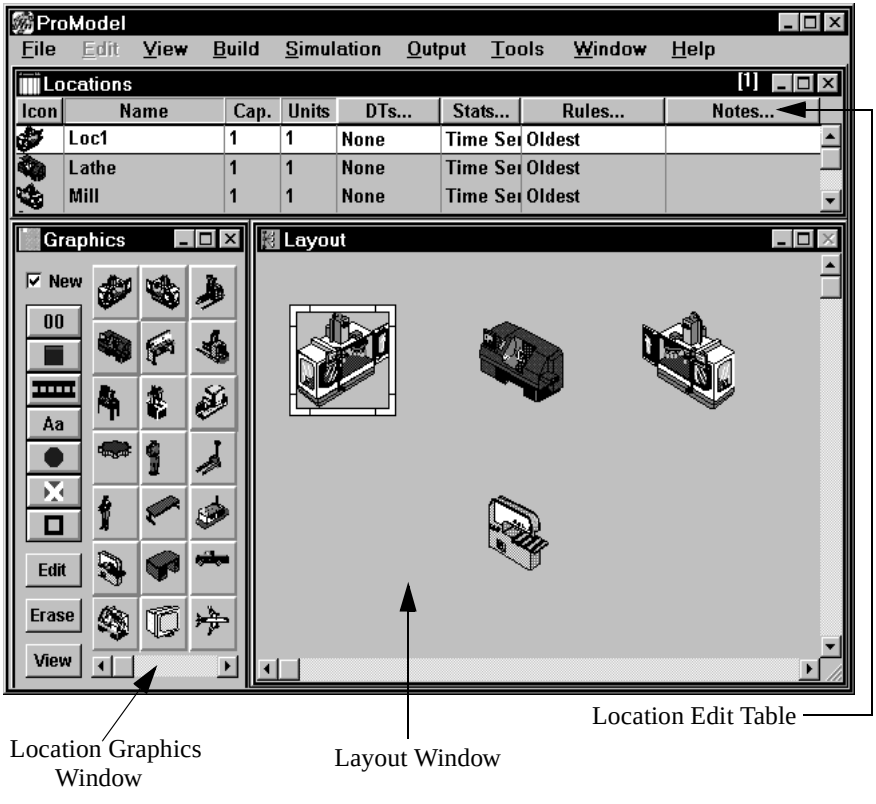
- Select **Locations** from the **Build** menu. The Locations Editor appears.

or...

- Right click on the existing location and select **Edit**.

7.1.1 Locations Editor

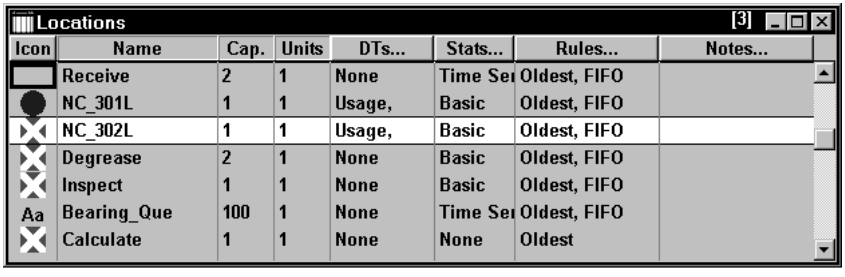
The Locations Editor consists of three windows: the Location Graphics window in the lower left portion of the screen, the Location edit table along the top of the screen, and the Layout window in the lower-right portion of the screen. These windows can be moved and resized using the mouse.



The Location edit table contains information about every location in the model including characteristics such as capacity and number of units. The Location Graphics window is a tool box used for creating, editing, and deleting locations graphically. Locations are positioned in the Layout window.

7.1.2 Location Edit Table

A location's characteristics can be modified with the Location edit table. The Location edit table contains fields for displaying the graphic icon, specifying the location name, and defining other characteristics of each location. Each of these fields is explained below. You can edit the desired field directly in some cases, or by selecting a record and clicking the column-heading button of the desired field.



Icon	Name	Cap.	Units	DTs...	Stats...	Rules...	Notes...
	Receive	2	1	None	Time Ser	Oldest, FIFO	
	NC_301L	1	1	Usage,	Basic	Oldest, FIFO	
	NC_302L	1	1	Usage,	Basic	Oldest, FIFO	
	Degrease	2	1	None	Basic	Oldest, FIFO	
	Inspect	1	1	None	Basic	Oldest, FIFO	
Aa	Bearing_Que	100	1	None	Time Ser	Oldest, FIFO	
	Calculate	1	1	None	None	Oldest	

Icon The graphic icon used to represent the location. Changing location graphics is done using the tools in the Location Graphics window. If multiple graphics have been used to define a location, the first graphic used is shown here. Clicking on the Icon button brings the graphic for the current location into view if it is not currently showing in the layout window.

Name The name of each location. Names can be up to 80 characters in length and must begin with a letter (for more information on naming items, see *Names on page 57* of the *ProModel Reference Guide*). A location's name can be changed by editing this field. The Search and Replace is automatically called when the name is changed.

Cap. The capacity of the location refers to the number of entities the location can hold or process at any one time. A location's maximum capacity is 999999. Entering INF or INFINITE will set the capacity to the maximum allowable value. If this field contains an expression, it will be evaluated at the start of the simulation before any initialization logic. Accordingly, a location's capacity will not vary during the simulation run.

Note Individual units of a multi-unit location may differ in capacity only if every unit's capacity is greater than 1. For example, in a location with two units, one may have a capacity of 5 and the other a capacity of 10. However, one unit may not have a capacity of one and the other a capacity of five. (See *Multi-Capacity, Multi-Unit, and Multiple Locations* on page 219.)

Units The number of units of a location, up to 999. A multi-unit location works like several locations with common characteristics. (See *Multi-Capacity, Multi-Unit, and Multiple Locations* on page 219.)

DTs Click on this heading button to define location downtimes, including any setup times. (See *Location Downtimes on page 221*.)

Stats Click on this heading button to specify the level of statistical detail to be gathered for the location. (To view a location's statistics after a simulation run, choose View statistics from the Output menu.) Three levels of data collection are available:

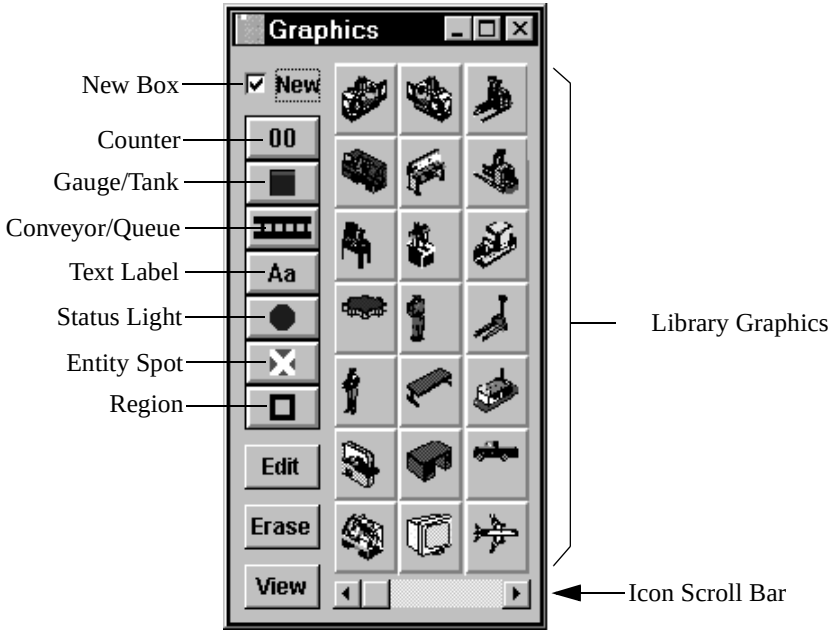
- **None** No statistics will be collected.
- **Basic** Only utilization and average time in location will be collected.
- **Time Series** Collects basic statistics and time series tracking the contents of the location over time.

Rules This field defines (1) how a location selects the next incoming entity from several that are waiting to enter this location, (2) how multiple entities at a location queue for output, and (3) which unit of a multi-unit location is selected by an incoming entity. (See *Rules Dialog Box on page 232*.) To edit any of this information at a location, click on the heading button to open the Location Rules dialog box.

Notes Enter any optional notes about a location in this field or click on the heading buton to open a larger Notes window.

7.1.3 Location Graphics Window

The Location Graphics window provides a graphical means for creating locations and changing their icons.



Icons added to the layout will either represent a new location or be added to an existing location's icon depending on whether the New box at the top right of the window is checked or unchecked.

New Mode

Allows you to create a new location record each time you place any location graphic on the layout. The new location is given a default name which may be changed if desired. New mode is selected by checking the New box [X] at the top of the Graphic Tools window.

Edit Button Displays the Library Graphic Dialog Box used to change the color, dimensions, and orientation of the location graphic.

Erase Button Erases the selected location graphic in the Layout window without deleting the corresponding record in the Location edit table.

View Button Brings the selected location in the edit table into view on the Layout window.

How To **Define a new location graphically:**

1. Check the **New** Box in the Location Graphics window.
2. Select a location symbol or icon from the Location Graphics window.
3. Click on the Layout window where you want the location to appear.
4. A new record is added automatically to the Location edit table. You may now change the default name to the desired location name.

How To **Define multiple locations, each with the same graphic:**

1. Check the **New** box inside the Location Graphic window.
2. Select the desired graphic.
3. While holding down the **SHIFT** key, click on the layout where each location should appear.

How To **Move a location graphic on the layout:**

- Drag the graphic to the desired spot on the layout.

How To **Move all graphics defined for a single location:**

- Drag inside the dashed box surrounding the graphic (do not drag on an individual graphic inside the box).

How To **Move multiple graphics for two or more locations at once:**

1. Click outside of any graphic and drag to create a rectangle encompassing all of the graphics to be moved.
2. Drag the rectangle to the desired position on the layout.

How To **Delete a location:**

1. Select the location record to be deleted in the Location edit table.
2. Select **Delete** from the **Edit** menu.

or...

1. Right click on the location graphic in the layout window.
2. Select **Delete *[location name]***.

How To **Erase a location graphic:**

1. Select the location graphic to erase.
2. Select the **Erase** button in the Location Graphics window or press the <Delete> key. The location graphic disappears, but the location record still exists in the Location edit table.

or...

1. Right click on the location graphic in the layout window.
2. Select **Delete Graphic**. The location graphic disappears, but the location record still exists in the Location edit table.

How To **Bring a location graphic into view that is off the layout:**

1. Highlight the record of the desired location in the Location edit table.
2. Select the **View** button in the Location Graphics window or click on the icon heading button.

Add Mode

Allows you to add additional graphics to an existing location, such as a text label, an entity spot, or a status light. A location with multiple graphics will be enclosed by a dashed box. Add mode is selected by *un*checking the New box [ ] at the top of the Graphic Tools window.

How To **Add an icon or symbol to an existing location:**

1. Uncheck the **New** Box in the Location Graphics window.
2. Select a location symbol or icon from the Location Graphics window.
3. Click on the Layout window where you want the additional icon to appear.
4. The graphic or symbol is added to the location.

7.1.4 Location Graphics

A location may have any one or more of the following graphics selected from the Location Graphics window.

Counter A counter representing the current number of entities at a location. The options available with counters are explained below.

Gauge A vertical or horizontal sliding bar showing the location's current contents during the simulation (shown as a percentage of the capacity). This graph will be updated constantly as a simulation runs. The options available with gauges are explained below.

Tank A vertical or horizontal sliding bar showing the continuous flow of liquids and other substances into and out of tanks or similar vessels. This continuous modeling capability can be combined with discrete-event simulation to model the exchange between continuous material and discrete entities such as when a liquid is placed in containers. You may also use this feature to model high-rate, discrete-part manufacturing.

Conveyor/Queue A symbol representing a conveyor or a queue. To create joints in a conveyor or queue, click on the conveyor or queue with the right mouse button. Drag the joints to achieve the desired shape. Right click on a joint to delete it. The options available with conveyors and queues are described below.

Label Any text used to describe a location. The label is initially synchronized with the name of the location and changes whenever the location name is changed. The name, size, and color of the text may be edited by double clicking on the label or selecting it and clicking on the edit button (See *Logic Builder on page 489*). Once the name on a label is edited it will no longer be automatically changed when the location name is changed.

Status Light A circle that changes color during the simulation to show the location's status. For a single capacity location, the states displayed are *idle/empty, in operation, blocked, down*, and *in setup*. For multi-capacity locations, the displayed states are *up* (operational) and *down* (off-shift, on break, disabled).

Entity Spot An assignable spot on the layout where the entity or entities will appear while at the location. While an entity is at a location, the entity's alignment spot (defined in the Graphic editor) will appear exactly on top of the location's entity spot, allowing the two graphics to align exactly as desired. A multi-capacity location will use as many entity spots as defined (in the order defined) up to the capacity of the location. Entities in excess of entity spots will continue to stack up on the last entity spot defined.

Region A boundary used to represent a location's area. A region may be placed in the layout over an imported background such as an AutoCAD drawing to represent a machine or other location. This technique allows elements in the imported background to work as locations.

Library graphic Any of the graphics appearing in the library graphic menu. Use the scroll bar to view all available graphics. Library graphics may be created or modified through the Graphic Editor. The name for the graphic, the default name of any location created with that graphic, can be saved in the Graphic Editor (see *Graphic Editor on page 515*).

How To

Edit a graphic already on the layout:

- Double click on the graphic

or...

1. Select the graphic.
2. Click on the **Edit** button inside the Location Graphics window.

or...

1. Right click on the graphic in the layout.
2. Select **Edit Graphic** from the menu.

Note

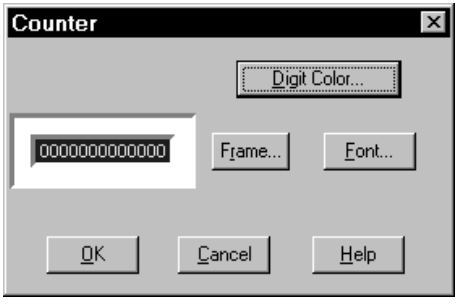
Location graphics notes:

1. Location graphics are painted on the layout in the order of the location list and, for any given location having multiple graphics, in the order that the graphic was added to the layout.

2. A location may include any of the above graphics and symbols. However, a location can have no more than one counter, one gauge, one tank, one queue, one status light, or one region.
3. Clicking on a layout graphic with no edit table on the screen displays the name of the element (location, etc.) represented by the graphic. With any edit table showing, hold down the CTRL key while clicking on the graphic to display the location name.

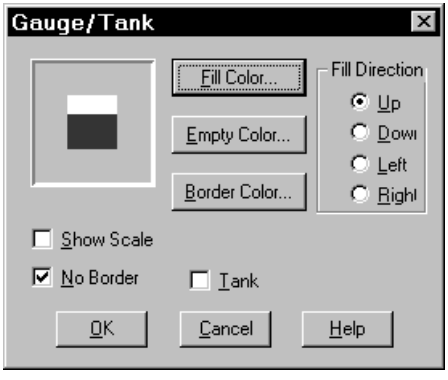
Counter Dialog Box

To edit the appearance of a counter, double click on the counter on the layout, select the counter and click on the Edit button, or right click on the counter and select edit. The counter dialog box allows you to choose the appearance of a graphic counter that is used to display the contents of a location. To change the digit color of the counter, click on the Digit Color button. To change the counter's background and border, click on the Frame button. The digit's font size and style may be changed by clicking on the Font button.



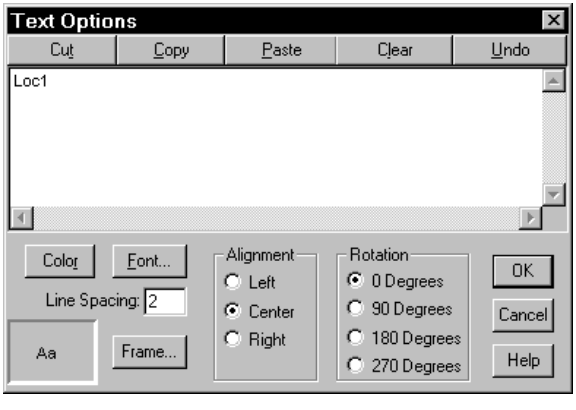
Gauge/Tank Dialog Box

When you create a gauge or tank, ProModel will prompt you to specify which type you wish to use before you paste it in the layout. To edit a gauge or tank on the layout, double click on the gauge or tank to display the gauge/tank dialog box, select the gauge/tank and click on the Edit button, or right click on the gauge or tank and select edit graphic. From the gauge/tank dialog, you may change a gauge to a tank and define its appearance, orientation, and fill direction. You may also access this dialog by selecting the gauge or tank and clicking on the Edit button.



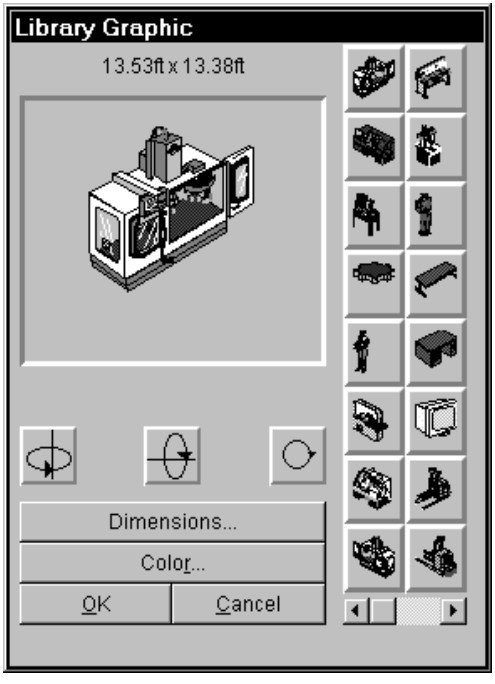
Text Dialog Box

To edit the appearance of a location label, you may double-click on the text once it is on the layout, select the text and click on the Edit button, or right click on the text and select edit. The text is typed in an edit window with several edit features available via buttons above the window. The font, color, alignment, rotation and frame may be changed from this dialog box. A sample of the currently chosen options is shown in the lower-left corner.



Library Graphic Dialog Box

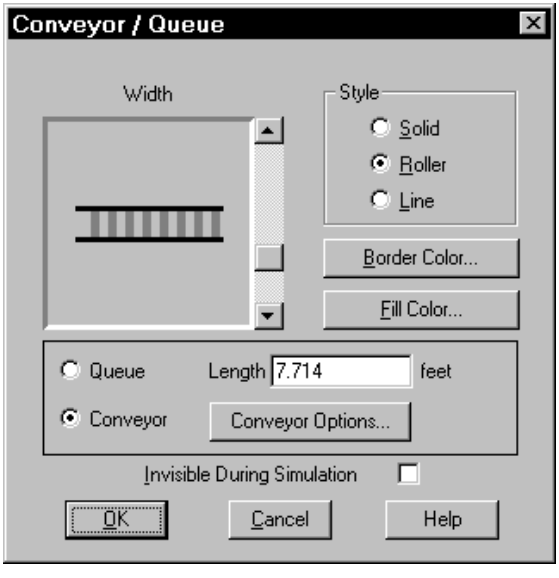
To change or edit a library graphic that represents a location, you may double click on the library graphic on the layout, select the graphic and click on the Edit button, or right click on the graphic and select edit.



This gives the option to change the icon, orientation, color, or graphic dimensions of the graphic. The default dimensions for the graphic, which are created in the Graphic Editor, are displayed above the graphic. To change the dimensions of the graphic, click on the Dimensions button. This gives you the ability to specify horizontal or vertical and feet or meters to change the graphic dimensions.

Conveyor/Queue Dialog Box

To control the look and operation of a conveyor or a queue, you may double click on the conveyor/queue in the Layout window, select the graphic and click on the Edit button, or right click on the conveyor/queue and choose edit graphic. The Conveyor/Queue dialog box appears. It also allows you to specify whether you wish to define the location as a conveyor or queue. Use the scroll bar to set the width of the conveyor or queue. Select the style by clicking on solid, roller (i.e., roller conveyor) or line. Click on the border color or fill color to change the color of the queue. If you want the queue to be visible only during edit time but invisible during run time, click on the Invisible During Simulation option. See the discussion on conveyors and queues later in this section for more information.



Conveyors

A conveyor is a location that simulates the movement of entities on either an accumulating or non-accumulating conveyor and appears with a conveyor graphic. Entities can only enter a conveyor at the beginning and leave at the end. For accumulating conveyors, if the lead entity is unable to exit the conveyor, trailing entities queue up behind it. For non-accumulating conveyors, if the lead entity comes to a stop, the conveyor and all other entities stop. Entities on a conveyor may not be preempted by other entities.

The capacity assigned to a conveyor limits the number of entities that can access a conveyor. However, the cumulative total length or width of the entities on the conveyor cannot exceed the conveyor length. In fact, the utilization statistics for a

conveyor reflect the amount of space utilized on the conveyor throughout the simulation, not the number of entities occupying the conveyor. Unlike other locations, an entity is not routed to the conveyor until there is room at the beginning for the entity to fit, even if the conveyor has capacity to hold it.

ProModel executes operation logic for entities entering a conveyor as soon as they enter unless the logic follows a MOVE statement. If no MOVE statement is encountered, entities begin their move on the conveyor after processing any logic. If a MOVE statement is encountered, entity movement is initiated. Any logic defined *after* a MOVE statement is processed when the entity reaches the very end of the conveyor.

Move time on a conveyor is based on the length and speed of the conveyor, as well as the length or width of the entity. The move time for an entity on a conveyor is calculated using the following formula:

Time = (Conveyor Length - Entity Length or Width)/Conveyor Speed

And the percentage utilization is calculated using this formula:

$$\text{Util \%} = \frac{\sum_{\substack{all \\ entities}} \frac{t_c}{C_c}}{T}$$

Where t_c = the time the entity spent on the conveyor whether moving or not
C_c = the conveyor capacity for that entity
T = the total simulation time

Note Unlike queues, MOVE statements for conveyors may not include a move time. Processing logic executed at the end of the conveyor may contain any operation statement except for CREATE, SPLIT AS, UNGROUP, or UNLOAD. Additionally, the ACCUM, COMBINE, and GROUP statements are not allowed at the end of non-accumulating conveyors.

Due to the space limitations of a conveyor, certain operation statements at the beginning of a conveyor are invalid including ACCUM, COMBINE, CREATE, GROUP, SPLIT AS, UNGROUP, and UNLOAD.

The default conveyor length is determined by the graphic scale, although this may be overridden by entering a different length. When a conveyor is modified graphically, the length will automatically be recalculated based on the graphic scale unless you uncheck the “Recalculate path lengths when adjusted” option. You can access this option from the Tools menu under Options.

Conveyor Graphics Display

When you use conveyors and want the graphics to display properly on the conveyor with no overlapping and little space between entities, use the following:

Entity Orientation on Conveyor	Requirements
Width-wise	1. Entity width on conveyor should equal horizontal dimension. 2. Entity length on conveyor should equal vertical dimension.
Length-wise	1. Entity width on conveyor should equal vertical dimension. 2. Entity length on conveyor should equal horizontal dimension.

Conveyor Animation

The animation of entities traveling along conveyors is displayed according to the logical length or width of the entity, not the scaled length or width of the entity graphic.

How To

Define a conveyor graphically:

1. Select the conveyor/queue symbol from the Location Graphics window.
2. Left click on the layout where the conveyor should start.
3. Add bends (i.e., joints) to the conveyor by moving the mouse and left clicking.
4. Right click to end the conveyor.

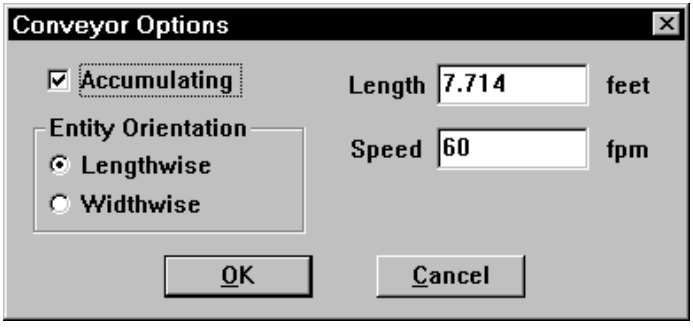
How To

Create bends in an existing conveyor:

1. Click on the conveyor with the right mouse button. From the menu that appears, select **Add Joint**. A small black square appears on the conveyor.
2. Using the left mouse button, drag the square in the direction you desire to bend.

Conveyor Options Dialog Box

The conveyor options dialog box is used to define the specifications for a conveyor. To access the conveyor options dialog box, you may double click on a conveyor, select the conveyor and click on the Edit button from the Location Graphics window, or right click on the conveyor and select edit. This opens the Conveyor/Queue dialog box from which the Conveyor Options dialog box can be opened by clicking on the Conveyor Options button. The Conveyor Options dialog box presents the following options:



Accumulating Select or deselect this option depending on whether the conveyor is to be accumulating or non-accumulating.

Entity Orientation Select Lengthwise or Width-wise depending on whether the entity is traveling on the conveyor in the direction of the entity length or in the direction of the width.

Length The length of the conveyor expressed in either feet or meters depending on the default specified in the General Information dialog.

Speed The speed of the conveyor in feet or meters per minute. The distance units can be set in the General Information dialog box.

Queues

A queue is a location that imitates the progressive movement and queuing of waiting lines. When an entity enters a queue, ProModel executes any operation logic for the entity and then moves it to the end of the queue. To have processing logic execute after an entity arrives at the end of a queue, use a MOVE statement in the operation logic. A MOVE statement causes the entity to move to the end of the queue where any additional operation logic defined will be processed. Operation logic following a MOVE statement actually gets processed after the elapsed time that the entity would have reached the very end of the queue if no other entities were ahead of it. Following a MOVE statement, any operation statement is valid except for CREATE, SPLIT AS, UNGROUP, UNLOAD, or another MOVE statement.

If a MOVE statement is specified that includes a move time (e.g., MOVE for 5.2 sec), the entity speed and length of the queue are ignored. If a move time is not included with the MOVE statement, the move time is based on the entity speed and length of the queue (if no queue length or entity speed is defined, the move time is zero).

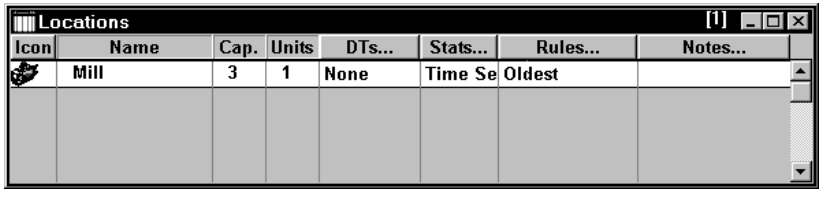
Entities in a queue may not be preempted by other entities and, once entities begin movement in a queue, are not allowed to pass each other. After the specified move time, however, entities continue processing any additional operation and output logic. A “No Queuing” rule specified for a queue location allows entities to depart in any order after completing their move time.

Queues are drawn from the beginning to the end of the center-line and are assigned a default length based on the graphic scale. However, the default queue length may be overridden by entering a different length. When a queue is modified graphically, the length will automatically be recalculated based on the graphic scale unless you checked the “Recalculate path lengths when adjusted” option. You can access this option from the Tools menu under Options.

7.1.5 Capacities and Units

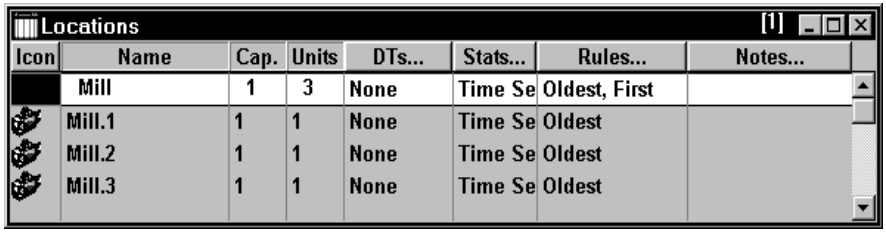
Capacities

A location *capacity* is the maximum number of entities it can hold at any one time. In general, multi-capacity locations are used to model locations such as queues, storage racks, waiting lines, ovens, curing processes, or any other type of location where multiple entities may be held or processed concurrently. Consider the following multi-capacity location:



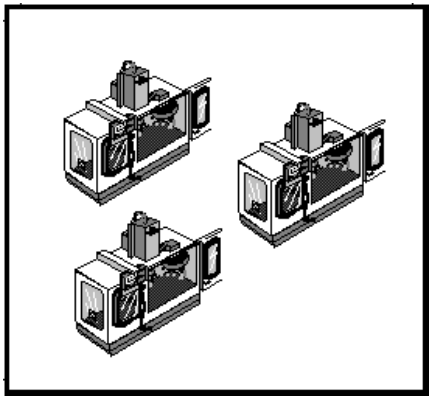
Units

A location *unit* is defined as an independently operating machine or station. When multiple, independently operating stations all perform the same operation and are interchangeable, they form a multi-unit location. Consider the following multi-unit location:



Multi-Capacity, Multi-Unit, and Multiple Locations

Sometimes it can be unclear whether to use multi-capacity, multi-unit, or multiple locations when defining parallel stations or machines. Suppose, for example, we have three parallel machines, each performing the same operation as shown below. There are three possibilities for defining the machines: (1) as a multi-capacity location, (2) as a multi-unit location, or (3) as multiple locations. The method you choose to define the locations depends on your application.



For many situations modeling parallel stations as a multi-capacity location works fine. By placing an additional graphic for each station, both the logic and visual effects of having parallel stations can be achieved. However, you should use a multi-unit location instead of a multi-capacity location when any of the following situations exist:

- Individual units have independent downtimes.
- It is important to collect individual statistics on each unit.
- It is important, for visual effect, to see entities select individual units by a routing rule other than First Available (e.g., By Turn, Fewest Entries, Longest Empty, etc.).
- It is important, for visual effect, to have a status light assigned to each unit.

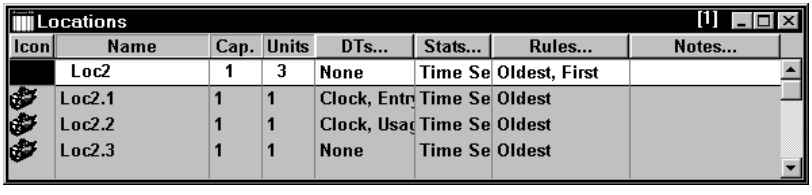
In some situations, it may even be desirable to model multi-unit locations as totally separate locations. Multiple locations should be used instead of multi-unit locations when:

- A path network is defined but each location must interface with a different node on the network.
- Different units have different processing times.
- The input for each unit comes from different sources.
- The routing is different for each unit.

Defining a Multi-Unit Location

To create a multi-unit location, enter a number greater than one as the number of units for a location. A corresponding number of locations will be copied below the multi-unit location record in the Location edit table, each with a numeric extension designating the unit number of that location. Successive graphics, representing individual units will be drawn to the right of the original location, but may be moved normally.

The original location record becomes the prototype for the unit records. Each unit will have the prototype's characteristics unless the individual unit's characteristics are changed. In the table below, each unit of the location has a clock-based downtime defined because the parent location, Loc2, was assigned a clock-based downtime. However, Loc2.1 has an additional entry-based downtime and Loc2.2 has an additional usage-based downtime. Any other characteristic, including the icon, can be changed as if the unit were an independent location.



Icon	Name	Cap.	Units	DTs...	Stats...	Rules...	Notes...
	Loc2	1	3	None	Time Se	Oldest, First	
	Loc2.1	1	1	Clock, Entr	Time Se	Oldest	
	Loc2.2	1	1	Clock, Usag	Time Se	Oldest	
	Loc2.3	1	1	None	Time Se	Oldest	

If the number of units is changed, the individual unit location records are automatically created or destroyed accordingly.

Individual units of a multi-unit location can be selected to process an entity according to the Selecting Incoming Entities option in the Rules dialog box. (See *Rules Dialog Box on page 232*.)

In the output report, scheduled hours for the parent location will be the sum of the scheduled hours for the individual units.

Note

Multi-Unit notes:

1. It is not possible to create a path network to interface with each unit of a multi-unit location. You must define the locations individually and use multiple locations as discussed above.
2. It is not possible to route an entity to a specific unit of a multi-unit location. For example, typing **Loc2.3** in the destination field of the Routing edit table is not allowed.

7.1.6 Location Downtimes

A downtime stops a location or resource from operating. A down resource or location no longer functions and is not available for use. Downtimes may represent scheduled interruptions such as shifts, breaks or scheduled maintenance. Or, they may represent unscheduled, random interruptions such as equipment failures. Downtimes may also be given preemptive or non-preemptive priority and may require one or more resources for repair times.

For single capacity locations, downtimes may be based on clock time, usage time, number of entities processed, or a change in entity type. Multi-capacity locations have only clock downtimes. If a downtime is occurring at a location and any other downtime starts (except a setup downtime), the two downtimes are processed together, not sequentially (i.e., downtimes overlap).

How To

Specify a location downtime:

1. Select the desired location in the edit table.
2. Click on the **DTs...** button. This brings up the downtime selection menu shown here.

Each selection opens an edit table for specifying the required elements of the downtime. Each edit table is described below.

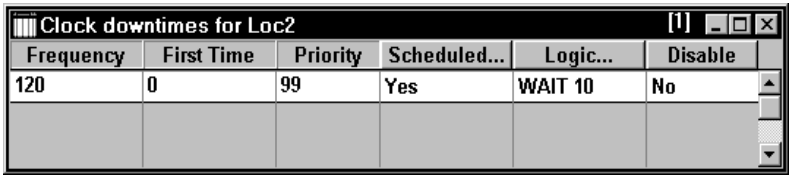
Hint

An alternative and more straightforward method for defining downtimes due to breaks or shifts is to use the Shift Editor. The Shift Editor also has the advantage of allowing a downtime to be defined for an entire group of locations.

Clock Downtime Editor

Clock downtimes are used to model downtimes that occur depending on the elapsed simulation time, such as when a downtime occurs every few hours, no matter how many entities a location has processed.

The Clock Downtime Editor consists of the edit table shown below. To access the Clock Downtime Editor, select Clock from the menu that appears after clicking the DT... heading button. Most expressions, including distributions, can be included in the Frequency, First Time, and Priority fields. (Consult *Appendix A Expressions & Statements* on page 259 of the *ProModel Reference Guide* to see if the specific function is valid in a particular field.)



Frequency The time between successive downtime occurrences. This option may be an expression. This field is evaluated as the simulation progresses, so the time between downtimes can vary.

First Time The time of the first downtime occurrence. If this field is left blank, the first clock downtime will occur according to the frequency field. This time is evaluated after any initialization logic.

Priority The priority (0-999) of the downtime occurrence. The default priority is 99, the highest non-preemptive priority.

Scheduled... Select YES if the downtime is to be counted as a scheduled downtime. Select NO if the downtime is to be counted as a non-scheduled downtime.

All scheduled downtimes will be deducted from the total scheduled hours reported in the output statistics and, therefore, will not be considered in computing utilization, percent down, etc.

Logic Enter any logic statements to be processed when the downtime occurs. When the logic has completed, the location becomes available. In the most simple case, the logic is simply a WAIT statement with a time value or expression which represents the duration of the downtime. Click on the heading button to open a larger edit window.

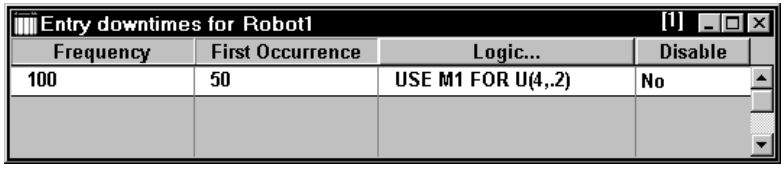
Disable Select YES to temporarily disable the downtime without deleting it from the table.

This example shows a simple, clock-based downtime where the location is down for 10 minutes every 2 hours (120 min). Because this time should not be included in the total scheduled or available hours, YES is selected in the “Scheduled” column.

Entry Downtime Editor

Entry downtimes are used to model downtimes when a location needs to be serviced after processing a certain number of entities. For example, if a paint machine needs to be refilled after painting every 100 cars, then an entry downtime should be defined. The downtime occurs *after* the entity that triggered the downtime leaves the location.

The Entry Downtime Editor consists of the edit table shown below. To access the Entry Downtime editor, select Entry from the menu that appears after clicking the DT... heading button. Entry downtimes are only available for single capacity locations. It contains fields for defining downtimes based on the number of entries that have been processed at a location. Most functions, including distributions, can be included in the Frequency and First Occurrence fields. (See *Appendix A Expressions & Statements* on page 259 of the *ProModel Reference Guide* to see if a specific function is valid in a particular field.)



Frequency The number of entities to be processed between downtime occurrences. This may be a constant value or a numeric expression and is evaluated as the simulation progresses.

First Occurrence The number of entities to be processed before the first downtime. This may be a value or a numeric expression. If left blank, the first downtime will be based on the frequency entered.

Logic Any logic statements to execute when the downtime occurs. Normally, this logic is simply a time expression representing the length of the downtime. Click on the heading button to open a larger edit window.

Disable Select YES to temporarily disable the downtime without deleting it from the table.

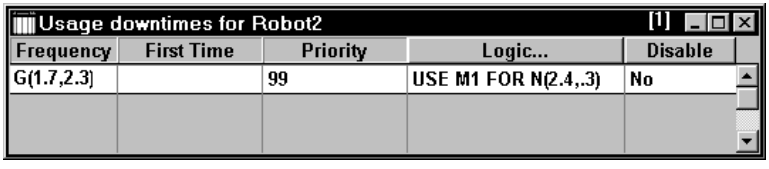
In the example above, Robot1 will go down every 100 entries, with the first downtime occurring after only 50 entries. When the downtime occurs, it will require a resource (M1) to service the machine for some amount of time between 3.8 and 4.2 minutes. If resource M1 is unavailable when requested, the robot will remain down until M1 becomes available.

Note Entry-based downtimes do not accumulate. For example, if a downtime cannot occur because the priorities of the entities being processed are at least 2 levels higher than the priority of the downtime, only the first downtime resumes after processing the entities. All others are ignored.

Usage Downtime Editor

Usage downtimes are used to model downtimes that occur after a location has been operating for a certain amount of time, such as if a machine fails due to tool wear after so many hours of operation. Usage downtimes are different from clock downtimes because usage downtimes are based on location operation time, which does not include blocked time. Clock downtimes are based on total elapsed simulation time which includes operation time, blocked time, idle time, etc. Usage downtimes are only available for single capacity locations.

The Usage Downtime Editor consists of the edit table shown below. It contains fields for defining location downtimes based on the actual time in use. Most functions, including distributions can be included in the Frequency, First Time, and Priority fields. (See *Appendix A Expressions & Statements* on page 259 of the *ProModel Reference Guide* to see if a specific function is valid in a particular field.)



Frequency The usage time between downtimes.

First Time The time in use before the first downtime occurrence. Leave blank if the first time is to be based upon the frequency entered.

Priority The priority, between 0 and 999 of the downtime. The default priority is 99, which is the highest non-preemptive priority. Generally, usage downtimes tend to be preemptive and should have priority values greater than 100.

Logic Any logic statements to be processed when the downtime occurs. Typically, this field contains a time expression representing the length of the downtime. Click on the heading button to open a larger edit window.

Disable Select YES to temporarily disable the downtime without deleting it from the table.

In this example, Robot2 will experience breakdowns according to a Gamma distribution with shape and scale parameters 1.7 and 2.3. Maintenance resource M1 will be used to service the robot. The repair time is normally distributed with a mean of 2.4 minutes and a standard deviation of .3 minutes.

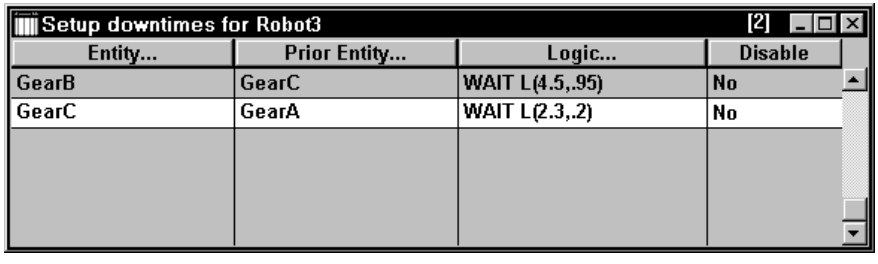
Note Usage-based downtimes do not accumulate. For example, if a downtime cannot occur because the priorities of the entities being processed are at least 2 levels higher than the priority of the downtime, only the first downtime resumes after processing the entity. All others are ignored.

Setup Downtime Editor

Setup downtimes should be used to model situations where a location can process different types of entities but needs to be setup to do so, such as a drilling station handling various parts, each requiring a different drill bits. Setup downtimes will not overlap, but will preempt other downtimes in a manner similar to that of an entity. Setup downtimes are only available for single capacity locations.

Note that a setup downtime is assumed to occur only when an entity arrives at a location and is different from the previous entity to arrive at the location. Consequently, the word ALL in the prior entity field means all *except* the same entity type.

The Setup Downtime Editor consists of the edit table shown below. It contains fields for defining location downtimes based on the arrival of a new entity type.



Entity The incoming entity for which the setup occurs. If the setup time for all entity types is identical when shifting from the same prior entity, the reserved word ALL may be entered.

Prior Entity The entity preceding the entity for which the setup occurs. If the setup is the same regardless of the preceding entity, you may enter the reserved word ALL.

Logic Enter any logic statements to be processed when the downtime occurs. Click on the heading button to open a larger edit window.

Disable Select YES to temporarily disable the downtime without deleting it from the table.

This example shows that the time to setup Robot3 depends on the arriving entity and the prior entity. If a GearB follows a GearC, the setup time for the machine will be based on a Lognormal distribution with a mean of 4.5 minutes and a standard deviation of .95 min. But if a GearC follows a GearA, the setup time will be based on a Lognormal distribution with a mean of 2.3 min and a standard deviation of .2 min.

Location Priorities and Preemption

Priorities determine which entity or downtime uses a location when more than one entity or downtime is contending for it. Priorities may be any value or expression between 0 and 999, with higher values having higher priority. For simple prioritizing, you should use priorities from 0 to 99. Priorities greater than 99 are used for preempting (bumping or displacing) entities or downtimes currently occupying a location.

Priority values are divided into ten levels (0 to 99, 100 to 199, ..., 900 to 999), with values beyond 99 used for preempting entities or downtimes of a lower priority level. Multiple preemptive levels make it possible to preempt entities or downtimes that are themselves preemptive. This means that an entity, EntA, with a priority of 99 can be preempted by another entity, EntB, with a higher priority level of 199. In turn, another entity, EntC, with a priority of 299 can preempt EntB at the same location.

To preempt an entity currently using a location, a preempting entity or downtime must have a priority at least ONE level higher than the entity currently at the location. To preempt a downtime in effect at a location, a preempting entity must have a priority at least TWO levels higher than the current downtime. Since all overlapping location downtimes are processed concurrently (except setup downtimes), a downtime cannot, in effect, preempt another downtime.

A preempted entity will resume processing where it left off unless the location was in the middle of a setup downtime. If the entity initiated a setup downtime before being preempted, it will begin processing the setup logic from the beginning when it resumes.

Assigning Priorities

An entity or downtime accesses a location based on its priority. An entity is assigned a priority for accessing a location in the Destination column of the Routing edit table. A downtime is assigned a priority in the appropriate Downtime edit table. The first of the following examples shows a priority of 100 assigned to EntA as it tries to claim Loc2. This priority is high enough to preempt any entity at the location having a priority less than 100. It is not high enough, however, to preempt any downtimes at the location.

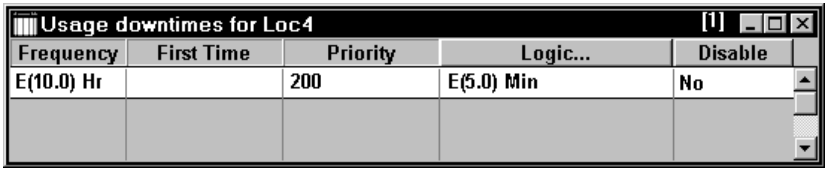
Process Table

Entity	Location	Operation (min)
EntA	Loc1	USE Res1 FOR N(3,.1)

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	**Loc2, 100**	First 1	MOVE FOR 1

This next example shows a priority of 200 assigned to a usage-based downtime at Loc4. This priority can preempt any entity at the location with a priority less than 200.



The following table shows the minimum priority level requirements for an incoming entity or upcoming downtime to preempt the current entity or downtime at the location.

Minimum Required Priority Levels for Preempting at a Location

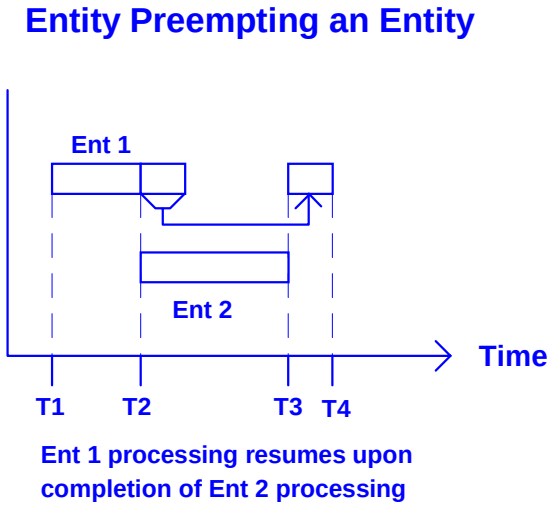
	To preempt the Current Entity	To preempt the Current downtime
Incoming Entity	1 priority level higher	2 priority levels higher
Upcoming Downtime	1 priority level higher	Downtimes overlap

- The upper left quadrant shows that for an entity to gain access to a location already processing another entity, the incoming entity must have a priority at least one level higher than the current entity's priority.
- The upper right quadrant shows that for an incoming entity to gain access to a location where a downtime is currently in effect, the entity must have a priority at least two levels higher than the downtime's priority.
- The lower left quadrant shows that a for a downtime to preempt an entity currently processing, the downtime must have a priority one level higher than the currently processing entity.
- The lower right quadrant shows that all location downtimes (except setup) are concurrent or overlapping. Setup downtimes preempt as if they were entities.

The following examples demonstrate the explanations above in greater detail.

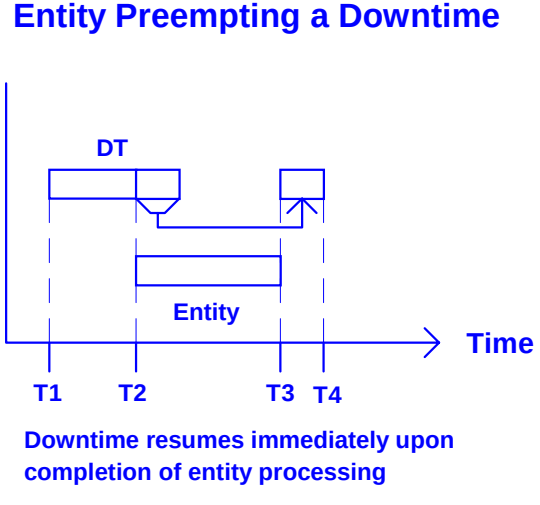
Example 1

The following example demonstrates what happens when Ent 1 with a priority of 99 is preempted by Ent 2 which has a priority of 100 or greater.



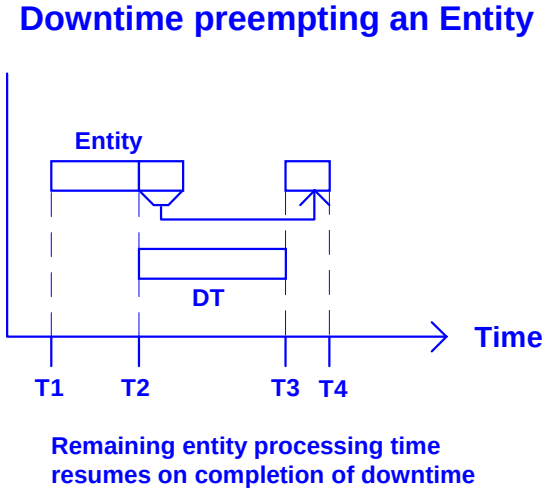
Example 2

This example demonstrates what happens when a downtime having a priority of 99 is preempted by an entity having a priority of 200 or greater.



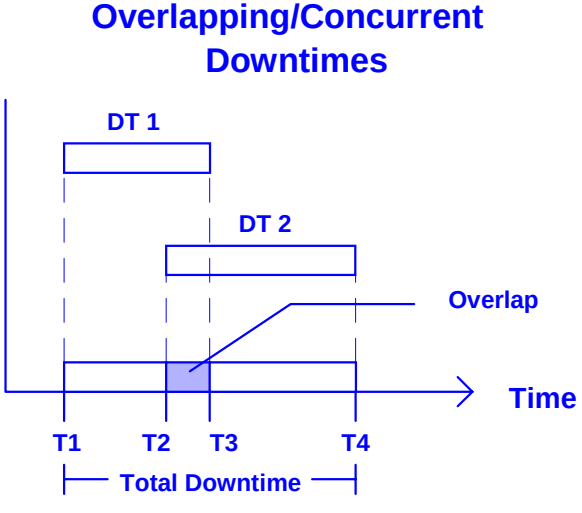
Example 3

This example demonstrates the behavior when an entity having a priority of 99 is preempted by a downtime with a priority value of 100 or greater.



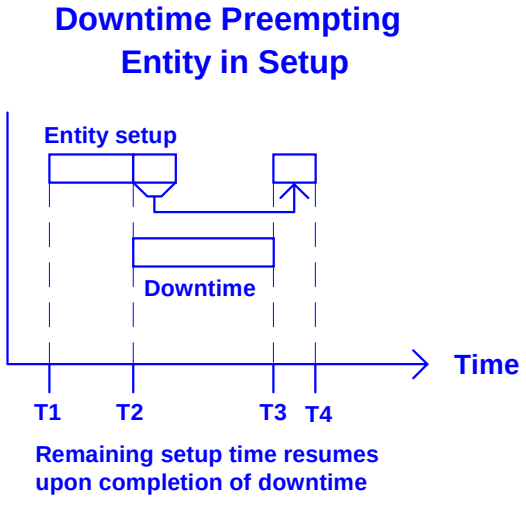
Example 4

This example illustrates how, regardless of the downtime priority values, downtimes will overlap. The exception is setup downtimes, which preempt downtimes exactly like entities (see Example 5).



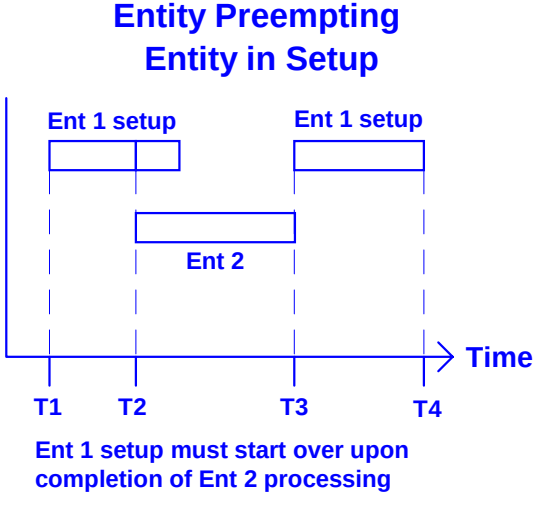
Example 5

This example demonstrates what happens when a setup downtime with a priority of 99 is preempted by a normal downtime having a priority of 100 or greater.



Example 6

This example demonstrates what happens when Ent 1 setup downtime with a priority of 99 is preempted by Ent 2 having a priority of 100 or greater.



Special Notes Regarding Location Downtimes

1. When an entity preempts another entity (Example 1), or when an entity preempts a downtime (Example 2), or when a downtime preempts an entity (Example 3), any resources owned by the preempted entity or downtime will be freed temporarily until the preempting entity or downtime finishes at the location. At that time, the original entity or downtime will seek to claim the *exact units* of the resource or resources it owned before the preemption occurred.
2. As shown in examples 5 and 6, an entity that requires a location setup will be treated differently depending on the preempting activity. If the preempting activity is another entity, the current setup in process will have to start over from the beginning. However, if the preempting activity is a downtime, the remaining setup time will finish upon completion of the preempting downtime.
3. Locations will not go down if they are in a blocked state. A location is blocked if it has an entity that cannot be routed because of the unavailability of the next location. This may also include the time an entity waits to enter a location based on a routing condition, such as LOAD.
4. Locations will not go down if any of the occupying entities are waiting for a resource or are waiting at any of the following statements:

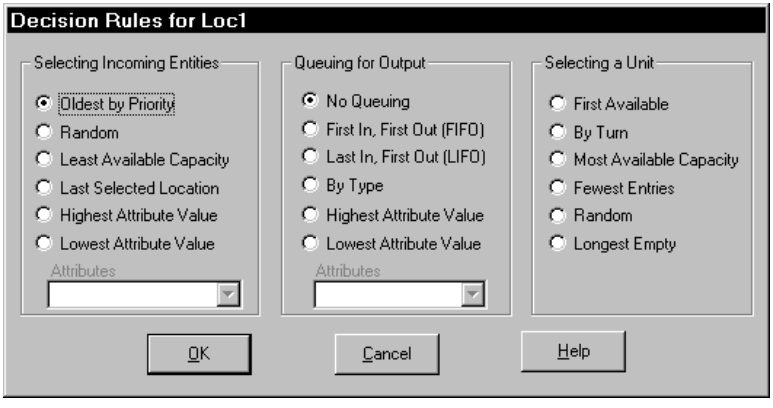
   WAIT UNTIL **ACCUM** **COMBINE** **MATCH**
   GROUP **JOIN** **LOAD**

5. In cases where a downtime or other entity attempts to preempt an entity's use of a location, a preemption process may be defined to override the default way of handling the preemption. See *Preemption Process Logic* on page 501.

7.1.7 Rules Dialog Box

The Rules dialog box, selected by clicking on the Rules button in the Locations edit table, is used to choose the rule for ProModel to follow when making the following decisions:

- Selecting incoming entities
- Queuing for output
- Selecting a unit



Selecting Incoming Entities

When a location becomes available and there is more than one entity waiting to enter, a decision must be made regarding which one to admit. The primary determining factor is the priority of the input routing. The entity with the highest routing priority will be admitted regardless of the incoming selection rule. However, if two or more entities have the same priority for claiming the location, then the location selects an incoming entity based on the incoming selection rules listed below.

Oldest by Priority Selects the entity waiting the longest among those having the highest routing priority.

Random Selects randomly with equal probability among all waiting entities.

Least Available Capacity Selects the entity coming from the location having the least available capacity. Ties are broken by the entity waiting the longest.

Last Selected Location Selects the entity coming from the location that was selected last. Ties are broken by the entity waiting the longest. If no entities are waiting at the last selected location, the *Oldest by Priority* rule takes effect.

Highest Attribute Value Selects the entity with the highest attribute value for a specified attribute. Ties are broken by the entity waiting the longest. Location attributes are also valid entries.

Lowest Attribute Value Selects the entity with the lowest attribute value for a specified attribute. Ties are broken by the entity waiting the longest. Location attributes are also valid entries.

Queuing For Output

When an entity finishes its operation at a location, other entities to finish ahead of it may not have departed. A decision must be made to allow the entity to leave or to wait according to some queuing rule. If one of the following queuing rules is not specified, "No Queuing" will be used.

No Queuing Entities that have completed their operations at the current location are free to route to other locations independent of other entities that have finished their operations. If this option is selected it is not displayed in the Rules Box.

First In, First Out (FIFO) The first entity completing operation must leave for its next location before the second entity completing its operation can leave, and so on.

Last In, First Out (LIFO) Entities that have finished operations queue for output LIFO so the last one finished is the first to leave.

By Type Entities that have finished their operations queue for output FIFO by entity-type so the routing for each entity type is processed independently of routings for all other types.

Highest Attribute Value Entities that have completed operations queue for output according to the highest value of a specified attribute.

Lowest Attribute Value Entities that have completed operations queue for output according to the lowest value of a specified attribute.

Selecting a Unit

If the location has multiple units, then incoming entities must select which available unit to use. One of the following rules may be selected. These decision rules apply to multi-unit locations only.

First Available Selects the first available unit.

By Turn Rotates the selection among the available units.

Most Available Capacity Selects the unit having the most available capacity. This rule has no effect with single capacity units.

Fewest Entries Selects an available unit with the fewest entries.

Random Selects an available unit randomly.

Longest Empty Selects the unit that has been empty the longest.

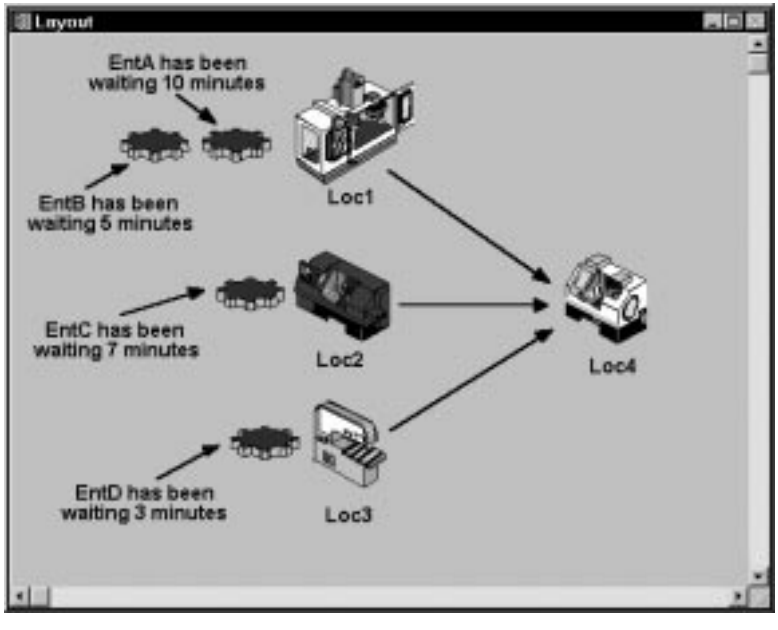
When specifying the decision rules for selecting incoming entities at a location, it is important to remember that the routing of an entity is also dependent on the queuing for output decision rules at the previous location. The following example will clarify this principle.

Rules Dialog Box Example

Consider a location, Loc1, which has a “Last In, First Out (LIFO)” as the queuing for output rule. Suppose that two other locations, Loc2 and Loc3, have “No Queuing” for the output rule. The three locations, Loc1, Loc2, and Loc3 feed into Loc4 which has an “Oldest by Priority” rule for selecting incoming entities.

Locations [1]

Icon	Name	Cap.	Units	DTs...	Stats...	Rules...	Notes...
	Loc1	2	1	None	Time Se	Oldest	
	Loc2	1	1	None	Time Se	Oldest	
	Loc3	1	1	None	Time Se	Oldest	
	Loc4	1	1	None	Time Se	Oldest	



Two parts are queued for output at Loc1. The part waiting the longest, EntA, at Loc1 has been waiting 10 minutes. The other part, EntB, which queued for output after EntA, has been waiting 5 minutes. At Loc2, the part queued for output, EntC, has been waiting 7 minutes. At Loc3, the part queued for output that has been waiting the longest, EntD, has waited 3 minutes.

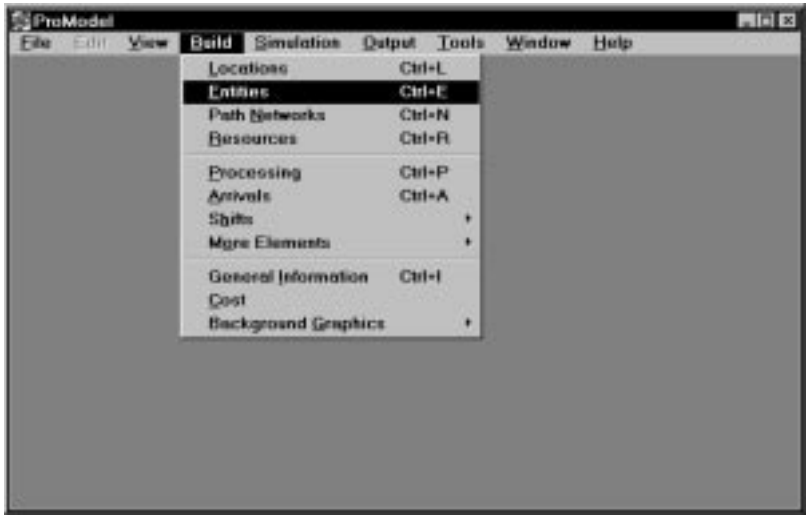
The part to enter Loc4 first is EntC at Loc2 which waited 7 minutes. Even though EntA has been waiting ten minutes, it must wait until EntB has been routed, because EntB is ahead of it in the output queue according to the LIFO queuing rule. Once Loc4 finishes processing EntC, EntB at Loc1 enters Loc4. EntB enters before EntA because entities must be output before a destination selects incoming entities. Next, EntA at Loc1 enters Loc4 after which EntD at Loc3 enters Loc4.

7.2 Entities

Anything that a model processes is called an “Entity.” Parts, products, people or paperwork should be modeled as entities. Entities may be grouped, such as when several boxes are stacked on a pallet (through the GROUP statement); consolidated into a single entity, such as when a tire is joined to a rim to form a wheel (through the JOIN statement); split into two or more entities, such as when a pipe is cut to a certain length (through the SPLIT AS statement); or converted to one or more new entities (through the RENAME or CREATE statement or by defining multiple outputs in the routing).

Each entity type has a name and a name index number. In logic and expressions, an entity can be referred to by name or by its name-index number using the ENT() function. The ENT() function allows a statement requiring an entity name to use an expression that may change to reference different entity names as a simulation progresses. See *Ent()* on page 145 of the *ProModel Reference Guide* for more information.

Entities may also have user-assigned attributes to represent such things as dimensions, weights, pass/fail status, group identifiers, etc.



How To

Access the Entities Editor:

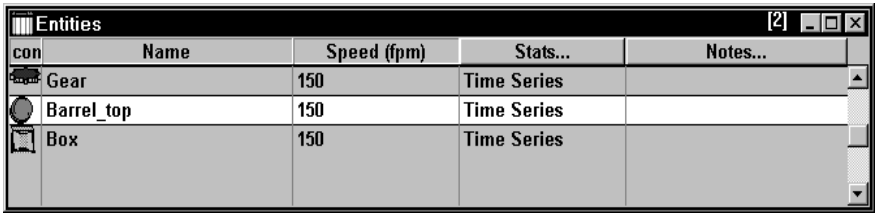
- Select **Entities** from the **Build** menu.

or...

- Right click on the existing entity and select **Edit**.

7.2.1 Entities Editor

Entity types are created and edited with the Entities Editor. The Entities Editor consists of (1) an edit table to define the name and specifications of each entity type in the system, and (2) the Entity Graphics window for selecting one or more icons to represent each entity. The fields of the edit table are explained below.



Icon This is the graphic icon used to represent the entity during the animation. Entity graphics are defined or modified using the Entity Graphics window. This icon can vary during the simulation. See *Defining Multiple Entity Graphics on page 241*.

Name The entity name. See *Language Elements on page 56* the *ProModel Reference Guide* for more information on naming.

Speed This entry is optional and applies to self-moving entities such as humans. It defines the speed in feet or meters (depending on the distance units chosen in the General Information Dialog box) per minute to be used for any of the entity's movement along a path network. When creating a new entity, a default value of 150 fpm (or 50 mpm for metric systems) is automatically entered. This is roughly the speed of a human walking.

Stats The level of statistical detail to collect for each entity type: None, Basic, or Time Series. Time series statistics must be selected if you wish to view a time series plot in the output module.

Notes Any information you wish to enter about the entity, such as material, supplier name, etc.

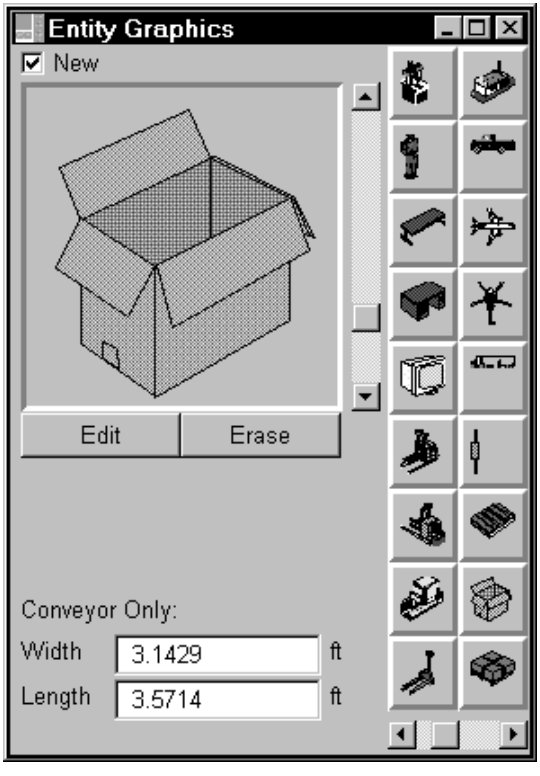
7.2.2 Defining Entities

Entities are typically defined graphically by clicking on a desired library graphic in the Entity Graphics window. Alternatively, you may define entities by simply entering their names and characteristics in the Entity edit table. Entity graphics are optional.

How To

Define entities graphically:

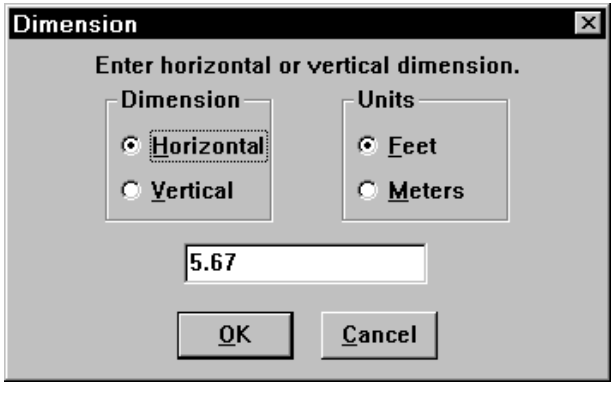
1. Select **Entities** from the **Build** menu.
2. Check the **New** box in the Entity Graphics window.
3. Select an icon for the entity. (Use the Graphic Editor to create new icons.)
4. Edit the name and other default entries for the entity in the Entity edit table.



7.2.3 Entity Graphic Dimensions

An entity has two sets of dimensions, a logical (length and width) dimension, and a graphical (horizontal and vertical) dimension. An entity's length and width are used to determine the number of entities that can fit on a conveyor, and do not affect the size of the graphic on the screen during a simulation. They are changed in the fields labeled Length and Width in the Entity Graphics window. If multiple graphics are defined for an entity, each graphic can have a different length and width. Which side a user chooses to call the length or width is unimportant as long as the proper side is referenced when defining a conveyor. If no conveyors are defined in the model, no specifications of a length and width are necessary.

An entity's horizontal and vertical dimensions are used to determine the size of the graphic on the screen. These dimensions can be changed in two ways. The scroll bar to the right of the graphic will scale the graphic. In addition, the horizontal and vertical dimensions can be changed by clicking on the Edit button, then clicking on the Dimensions button from the resulting dialog box. The default dimensions are determined when an icon is created to scale in the Graphic Editor. If the size is changed using the scroll bar, the change will be reflected in the dimensions listed. If you change either the horizontal or vertical dimension from the dialog box, the size of the icon will change accordingly.



Note Since the horizontal and vertical dimensions must remain proportional, only one of the dimensions needs to be changed. The other dimension changes automatically to maintain proportionality.

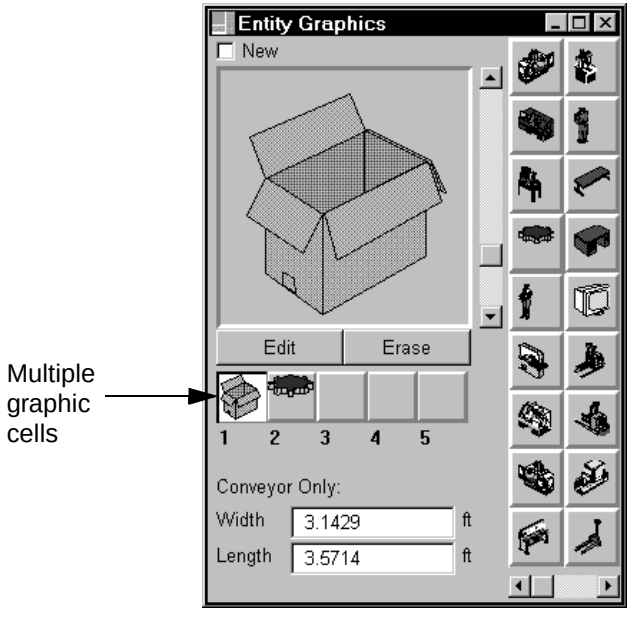
7.2.4 Defining Multiple Entity Graphics

Entity types can be assigned more than one graphic to represent the entity at various stages of production or traveling in different directions. An entity representing a bike could be assigned two graphics, one without the wheels attached, the other with the wheels attached. During the simulation, when the wheels get attached with a JOIN statement, the graphic representing the bike could be changed from the one without wheels to the one with wheels using the GRAPHIC statement (see *Graphic* on page 160 of the *ProModel Reference Guide* for information).

How To

Define multiple graphics for an entity type:

1. *Uncheck* the **New** box on the Entity Graphics window. Numbered graphic cells appear in the Entity Graphics window.



2. Click on the desired cell.
3. Select a library graphic from the graphics menu.
4. Repeat steps two and three until all the desired graphics have been assigned to the entity type.

The graphic that represents an entity during a simulation will be the first in this series until an entity's graphic is changed with the GRAPHIC statement.

7.2.5 Preemptive Entities

Often during a simulation, it is desirable to have an entity preempt an activity at a location or resource in order to gain access to that location or resource. These situations can be modeled using preemptive priorities. An entity with a high enough priority can take over a location processing an entity or a location that is down. An entity with high enough priority can also take over a resource when it is being used by another entity or when it is off shift. When an entity takes over a location that was down or in use by another entity, the entity has preempted the downtime or the other entity.

In a multi-capacity location, the occupying entity will be preempted only if there is no more capacity at the location and the occupying entity is undergoing an operation time. Further, the occupying entity cannot be one that has been split, created, grouped, combined, ungrouped or unloaded at the location.

An entity must have a priority one level higher than an occupying entity to preempt the occupying entity. An entity must have a priority that is two levels higher than a downtime to preempt the downtime. If an entity does not have a high enough priority to preempt another entity or downtime at a location, it waits in line (oldest by priority) to access the location (see *Location Priorities and Preemption* on page 226).

Note that the priority of an entity is not defined for the entity itself. For claiming a location, it is defined in the destination field of the routing. For capturing a resource it is defined as part of the GET, JOINTLY GET, or USE statement. A priority may, however, be assigned to an attribute of a referenced entity when it attempts to access a resource or location.

Example of Preemptive Entities

In this example entity (EntA) arrives at location Loc1. Immediately upon arrival it requests to use resource Res1 for a normally distributed amount of time. The priority for obtaining the resource is 99, which means that it is a non-preemptive request. When Res1 becomes available, EntA will be first in line because it has the highest non-preemptive priority possible. When processing is complete for this entity, it is routed to Loc2 with priority 200. This means that it *can* preempt another entity or a downtime that may already be in process at Loc2. (See *Location Downtimes on page 221* and *Resource Downtimes* on page 274 for more details on entity preemption.)

Process Table **Routing Table**

Entity	Location	Operation (min)	Blk	Output	Destination	Rule	Move Logic
EntA	Loc1	Use Res1,99 For N(3,.1)	1	EntA	Loc2,200	First 1	MOVE FOR 2.5

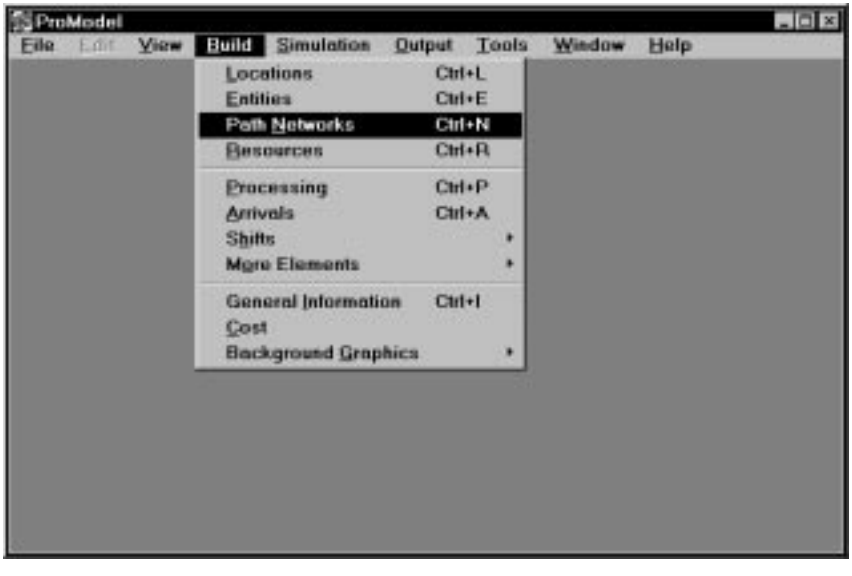
7.3 Path Networks

When resources are modeled as dynamic resources which travel between locations, they follow path networks. Entities moving by themselves between locations may also move on path networks if referenced in the move logic of the routing. Otherwise, they follow the routing path. Multiple entities and resources may share a common path network. Movement along a path network may be defined in terms of speed and distance, or simply by time. See discussion on Automatic Time and Distance Calculation, later in this section, for more information about movement according to speed and distance or by time.

There are three types of path networks: **passing**, **non-passing**, and **crane**. A passing network is used for open path movement where entities and resources are free to overtake one another. Non-passing networks consist of single-file tracks or guide paths such as those used for AGVs where vehicles are not able to pass. Crane networks define the operating envelope and interface points for bridge cranes only.

Passing and non-passing networks consist of nodes, which are connected by path segments. Path segments are defined by a beginning and an ending node and may be unidirectional or bi-directional. Multiple path segments, which may be straight or jointed, may be connected at path nodes. Crane networks consist of a crane envelope and nodes defining move positions. For all path networks, path nodes define the points where the resources using the network interface with processing locations.

Path Networks are defined in the Path Networks Editor, accessed from the Build menu.



How To **Create or edit a path network:**

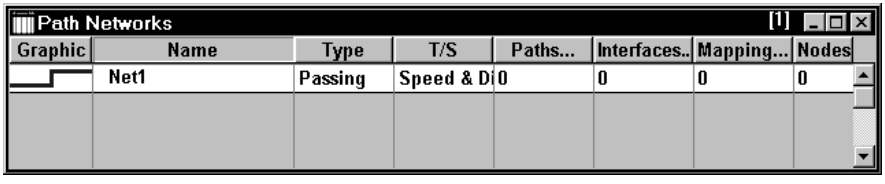
- Select **Path Networks** from the **Build** menu.

or...

- Right click on the existing path network and select **Edit**.

7.3.1 Path Networks Editor

The Path Networks Editor consists of an edit table with fields for defining basic information about each network, such as the network name, the type of network (Non-Passing, Passing, or Crane), and the basis for movement along the network (Speed and Distance or Time). Clicking on the appropriate heading button will bring up a table for defining nodes, path segments, and location node interfaces.



The following explains each field of the Path Networks edit table.

Graphic For passing or non-passing path networks, this button displays the Path Color dialog, which allows you to define the color of the path network. For crane path networks, it displays the Crane Graphic edit dialog, which allows you to specify the colors, bridge separation, and graphical representation of the crane. Click on the heading button or double click in this field to bring up the graphic dialog. Both dialogs allow you to specify whether or not the network will be visible at run time.

Name A name that identifies the path network. For more information about valid names, see *Names on page 57* of the *ProModel Reference Guide.*

Type Set this field to Non-Passing if you want entities and resources to queue behind one another on the path network. If a path is Non-Passing, entities may not pass each other, even if an entity is traveling at a faster speed than the one in front of it. Set this field to Passing if you want entities or resources to pass each other on the path network. If you want to create a crane path network, select Crane.

T/S Set to either **Time** or **Speed and Distance** as the basis for measuring movement along the network. See the discussion on Automatic Time and Distance Calculation later in this section for more information. This option is not available for crane path networks.

Paths The number of path segments in the network. Clicking on the heading button opens the Path Segment edit table where the user may define the network's node to

node connections. The Path Segment edit table is covered in more detail later in this section. This option is not available for crane path networks.

Interfaces The number of location-node interfaces in the path network. If an entity will be picked up or dropped off at a particular location by a resource, that location must connect to a node through a location-node interface. Clicking on the heading button opens the Interfaces edit table where the user may define nodes that connect to processing locations. The Interfaces edit table is covered in more detail later in this section.

Mapping The number of entries in the Mapping edit table. Clicking on the heading button opens the Mapping edit table where the user may map destinations to particular branches of the network. (The Mapping edit table is covered in more detail later in this section.) This option is not available for crane path networks.

Nodes The number of nodes defined in the Nodes edit table. Nodes are created automatically when graphically defining path segments. Click on this heading button to open the Node edit table, which may be used to define nodes manually or set Node Limits on one or more nodes. Nodes may also be used to control a resource's behavior through node logic or search routines such as work and park searches (see *Resources on page 261*). The Nodes edit table is covered in more detail later in this section. This also indicates the origin, rail, bridge, and additional nodes for the crane.

How To Create a Path Network graphically:

1. Set the default time and distance values per grid unit from the Grid dialog box.
2. Choose **Path Networks...** from the **Build** menu.
3. Enter the name of the network in the Path Networks edit table.
4. Select either **Passing**, **Non-passing** or **Crane** as the network type. (See cranes explained further in this section.)
5. Select either **Speed and Distance** or **Time** as the travel basis.
6. Click on the **Paths...** heading button to open the Path Segment edit table.
7. Lay out the network using the mouse buttons as described below.

How To Create path segments:

1. Left click to create a node and begin a path segment.
2. Additional left clicks produce path joints.
3. A right click ends the segment and creates a new node.

How To **Modify path segments:**

To create a new path from an existing node

- Left click on that node.

To delete a joint

- Right click on an existing joint.

To add a joint

1. Right click anywhere on a path segment.
2. Select **Add Node** from the menu.
3. Drag the joint to the desired position.

Note If you hold down the CTRL key and move the cursor over a path segment or joint, the Add/Delete joint cursor will appear. From here, left-click to add or delete a joint.

To highlight a path on the layout and in the Path Segment edit table

- Left click on that path.

How To **Create additional nodes or move existing nodes:**

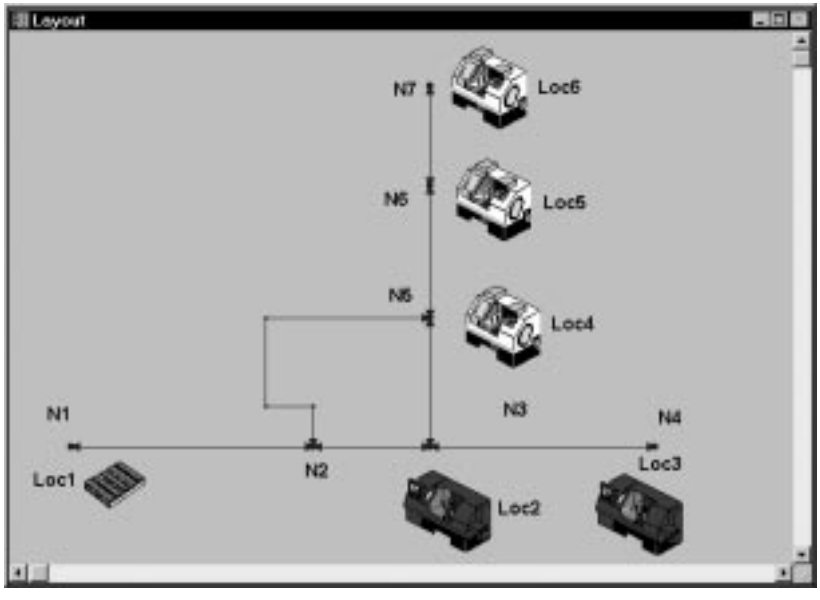
1. Click on the Nodes heading button in the Path Networks edit table.
2. Click the left mouse button to create a node on both the layout and in the Nodes edit table.
3. Drag an existing node to move that node.

How To **Move a path network:**

1. Click on the Paths heading button in the Path Networks edit table.
2. Left click on any path segment and drag to the desired position. The entire network will move.

7.3.2 A Typical Path Network

The following diagram shows a path network consisting of seven nodes (N1 to N7) connected by path segments. Path segments may be straight lines from one node to another, as in the segment from node N7 to node N6, or they can have any number of joints, such as the segment from node N2 to node N5.



7.3.3 Path Segment Edit Table

This table is used to define the Path Segments that make up a path network. When specifying travel according to time between nodes, the heading “Distance” changes automatically to “Time.”

Paths [1]

From	To	Bi	Time
N1	N2	Bi	17.16
N2	N3	Bi	41.23
N3	N4	Bi	11.22
N4	N5	Bi	40.10
N5	N6	Bi	18.95
N5	N2	Bi	42.20
N1	N7	Bi	14.64
N7	N6	Bi	21.30

The following defines the fields of the Path Segment edit table.

From The beginning node of the path segment.

To The ending node of the path segment.

BI Set to **Uni**-directional or **Bi**-directional depending on whether traffic can travel in only one or either direction.

Time If travel along the network is to be measured in time rather than in speed and distance, then enter the time required for a resource or entity to traverse the path segment. This value may be any numeric expression except for resource and downtime system functions. When travel along a path is measured in time, all resources and entities traveling along the path take the same amount of time to travel it, regardless of their speed. This field's title changes to “Distance” if the T/S field in the Path Networks edit table is set to Speed and Distance.

Distance If travel along the network is to be measured in terms of speed and distance, enter the length of the segment which determines the travel time along the path in conjunction with the speed of the resource or entity.

The value entered may be any numeric expression except for attributes, arrays, and system functions. This expression is evaluated only when the simulation begins.

The distance may be followed by a comma and a speed factor between .01 and 99. This speed factor may be used to model any circumstance affecting the speed of items traveling the path. For example, a resource may normally travel at 150 fpm, but may slow down as it goes around a corner to 80% of the original speed, 120 fpm. This would be entered as 100, .8 for a path segment 100 feet long which traversed the corner. This field's title changes to “Time” if the T/S field in the Path Networks edit table is set to Time.

Note

Path segment editing notes:

1. If no path segments have been defined for a network, resources and entities will move from node to node in zero time. See *Processing* on page 295 for more information about the Routing Move dialog box.
2. To move nodes already defined on the layout, click on the Nodes button and move the desired nodes.
3. To insure that all nodes can be seen by the user, two nodes cannot be located at the same point.

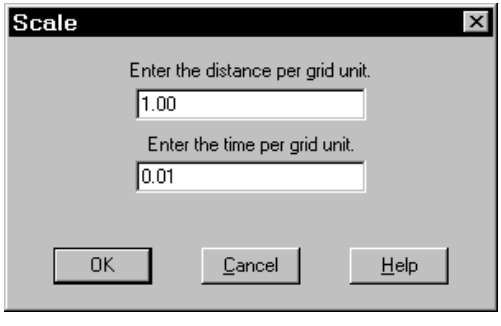
Automatic Time and Distance Calculation

The distance between two successive nodes or the time required to traverse a segment between two successive nodes is calculated according to the number of grid units between the nodes and the default time and distance values per grid unit. ProModel then automatically enters this time or distance in the Time/Distance column of the Path Segments edit table. Although the calculated time or distance may be edited later, relying on the automatic time and distance calculation feature allows path networks to be built to scale and saves time when defining path networks graphically. The time or distance for a path is automatically recalculated whenever the path is edited (lengthened or shortened) unless you unchecked the "Recalculate path lengths when adjusted" box under Options in the Tools menu.

How To

Set the default time and distance values per grid unit:

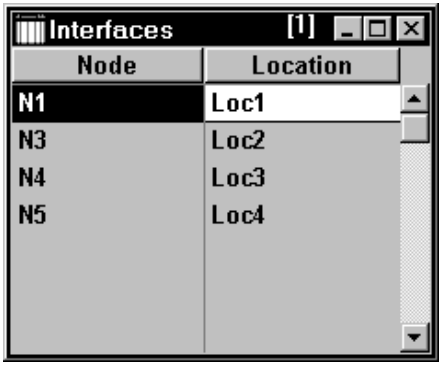
1. Select **Layout Settings** from the **View** menu.
2. Select the **Grid Settings** button.
3. Click on the **Scale...** button in the Grid dialog box.
4. Enter the time and distance values as shown below.



Note To set these values as defaults, you must check the Save as Default Grid Settings option on the grid dialog box.

7.3.4 Interfaces Edit Table

If an entity will be picked up or dropped off at a particular location by a resource, that location must connect to a node through a location-node interface. The Interfaces edit table is used to define location-node interfaces. The graphic below shows how to set node N1 to interface with location Loc1, node N3 to interface with Location Loc2, and so on, as in the example at the beginning of this section.



The fields of the Interfaces edit table are described as follows.

Node The node name.

Location The name of any locations which interface with the node. Nodes can interface with several locations, but a location may interface with only one node on the same path network.

How To

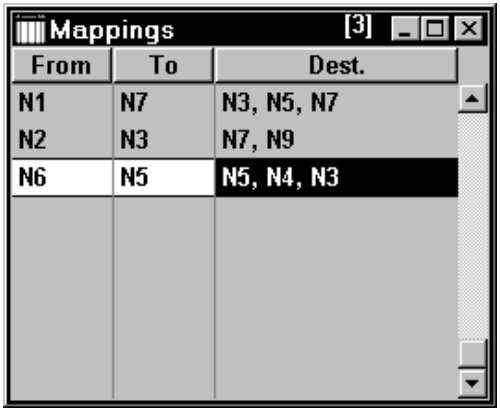
Create location-node interfaces

1. Left click on the desired node to begin rubber-banding a link or interface.
2. Left click on the desired location to complete the interface.

Note A node on a path network *may not* interface with a particular unit of a multi-unit location (i.e., Loc1.2). A node may only interface with the "parent" location (i.e., Loc1) of a multi-unit location.

7.3.5 Mapping Edit Table

If there are multiple paths emanating from one node to another node, the *default* path selection will be based on the shortest distance for speed & distance networks, and the least number of nodes for time based networks. These defaults can be overridden by explicitly mapping some destination nodes to specific branches that entities and resources will take when traveling out of a "from" node.



The fields of the Mapping edit table are described as follows.

From Entities and resources traveling out of this node will use this mapping record to decide which of the alternate branches will be taken next.

To The "from" node and the "to" node together define the branch to be taken next. This might also be interpreted as the node which entities and resources will go *through*, to reach one of the destination nodes.

Dest. Entities and resources whose *ultimate destination* is one of these nodes will be forced to take the branch that directly connects the "from" node to the "to" node.

How To

Create mappings using the Mapping edit table:

1. Click on the **Mapping...** heading button in the Path Network edit table. This will open the Mapping edit table.
2. Click on the **From** heading button and select the node to be mapped.
3. Click on the **To** heading button and select the terminating node for the branch to be mapped.
4. Click on the **Destination** heading button and select the desired node(s).

How To **Create mappings graphically:**

1. Click on the **Mapping...** heading button in the Path Network edit table. This will open the Mapping edit table.
2. Click on the *from* node in the Layout Window. This places the selected node in the From field.
3. Click on the *to* node in the Layout Window. Note that the *to* node must be directly connected to the *from* node with a single branch.
4. Click on the *destination* node(s) in the Layout Window. This places the selected node(s) in the Destination field.

Note ProModel automatically calculates and uses the shortest paths on unmapped portions of networks (if the network is time based, the path having the least number of nodes is used). Explicitly indicating shortest paths using mapping constraints will speed up the translation process, especially for models with complex networks.

An example of mapping two branches of a network is given on the following pages.

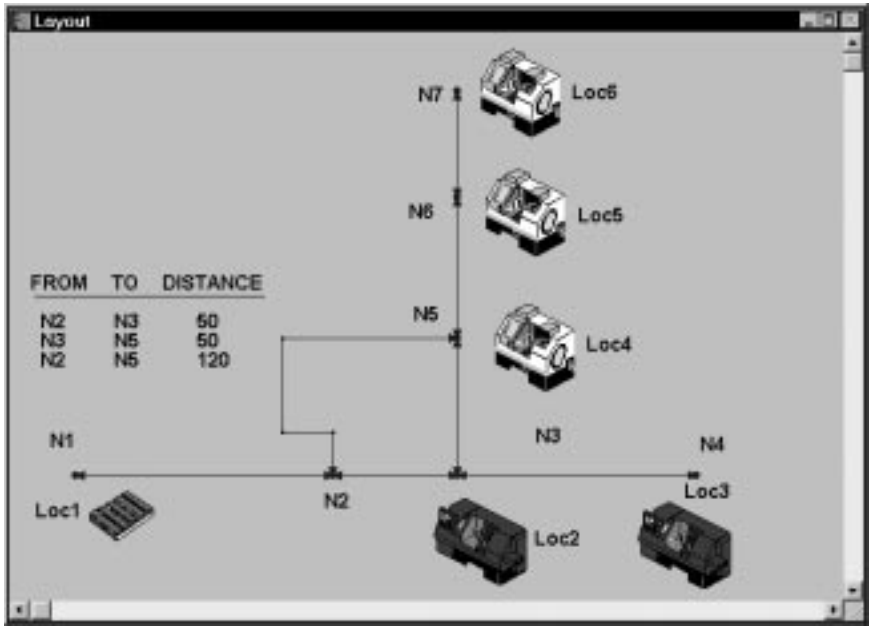
Mapping Example

The following example uses a path network diagram to demonstrate mapping.

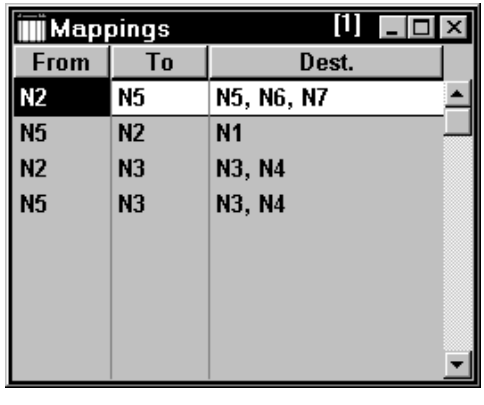
In this example, we wish to force resources and entities enroute from Loc1 to Loc4, Loc5, or Loc6 to take the branch directly connecting node N2 and node N5 to avoid traffic congestion at the intersection of the two main branches at node N3. Since there are multiple ways to go from N2 to N5, a decision as to which alternative will be used has to be made at N2.

In addition, we want resources and entities to follow the same path in the opposite direction when enroute from Loc4, Loc5, or Loc6 to Loc1. In this case, the decision must be made at N5.

Because the combined length of segments connecting N2 to N3 and N3 to N5 is shorter than the length of the single segment from N2 to N5, resources and entities based on speed and distance will normally take the former path to travel. To force them to take the longer path, we must specify mapping constraints.



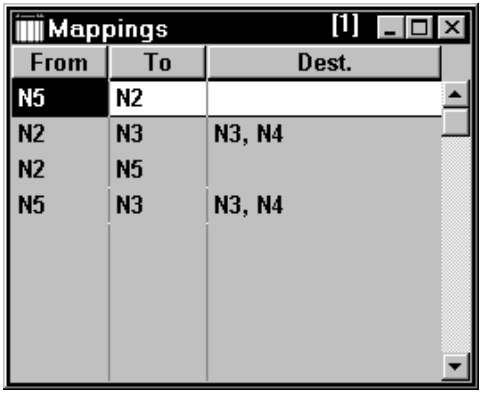
This case requires two explicit mapping constraints to override the selection of default paths in each direction: The first table entry forces entities and resources en-route from Loc1 to Loc4, Loc5, or Loc6 to override the default path and take the direct branch from N2 to N5. The second table entry forces entities and resources traveling from N5 (originally from Loc6, Loc5 or Loc4) to Loc1 to take the direct branch from N5 to N2. Entries 3 and 4 are *optional*, and might be useful to speed up translation, since the restrictions they impose allow the shortest path calculations to be bypassed.



There is a shortcut to force the same non-default path selection constraint to a number of destination nodes: For instance, if the vertical arm of the path network extended up

to include many other nodes N8, N9, etc., and locations Loc7, Loc8, etc., then we would change the Mapping edit table as follows:

1. Delete line 1 in the Mapping edit table.
2. Make sure that line 3 is there (it is *not* optional any more).
3. Include a line which reads: “From: N2, To: N5, Dest:<BLANK>.”

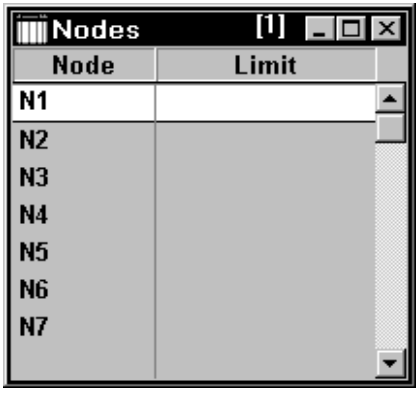


The empty destination column will be interpreted as “all other destination nodes” by ProModel.

Note For a “from” node (unless there is a branch map with a blank destination column), any nodes not explicitly listed in the destination columns of existing mapping records will be reached via the default path selections.

7.3.6 Nodes Edit Table

The Nodes edit table lists the nodes that make up a path network and is used to limit the number of resources and entities that may occupy a node at any given time. In addition to controlling traffic on a path network, nodes also define where resources interface with locations or where entities enter and leave the path network. Nodes may also be used solely to control a resource or entity's behavior through node logic or search routines such as work and park searches (see *Resources on page 261*).



The following defines the fields of the Nodes edit table.

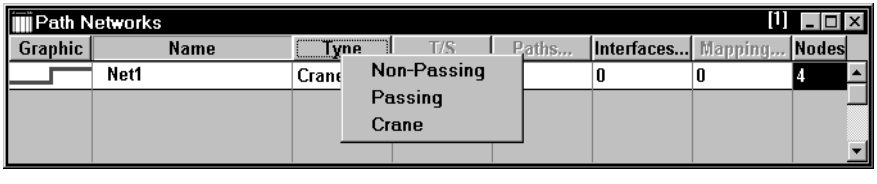
Name The node name.

Limit The maximum number of resources and entities that may occupy a node at any given moment. A blank entry means there is no limit.

7.3.7 Defining a Crane Envelope Path Network

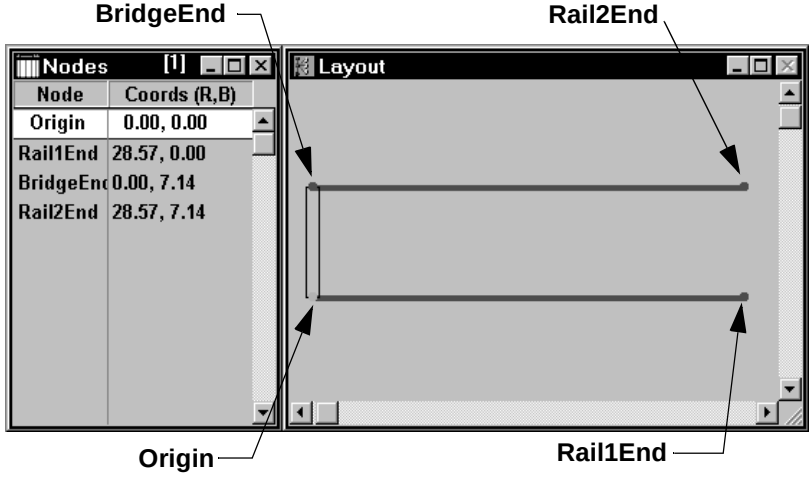
The first step in creating a crane in a model is to specify its operating envelope in the Path Networks module. The actual cranes to use a particular crane envelope are specified later in the Resources module.

When the Type button is clicked in the Path Networks edit table in ProModel, a menu appears as shown below. The third type of path network listed is the Crane.



How To Define a crane envelope (path network):

1. Select **Crane** as the network type from the path networks edit table. The following crane graphic appears in the Layout window. The horizontal lines represent the rails and the vertical line represents the bridge. The Nodes window also appears with four, pre-defined nodes (*coordinate axis values*) defining the crane envelope or coordinate system.



   All crane movement is defined relative to this R,B (Rail, Bridge) coordinate system. Initially, the origin is defined at the lower left corner of the envelope. You are free to adjust the crane envelope and move the origin to a different relative position. For example, you may want to position the rails vertically.

2. Click and drag the corner nodes to size and orient the crane envelope as desired. The rails always stay parallel and the same length. (Adjusting one rail makes the other rail follow.)

3. To move the entire crane envelope while the Nodes window is open, click on any part of the bridge or rails except the corner nodes and drag it to the desired position.

4. Edit the *coordinate axis values* in the Nodes window if necessary to reflect the actual dimensions of the crane envelope. The zero coordinates of these pre-defined nodes cannot be changed.

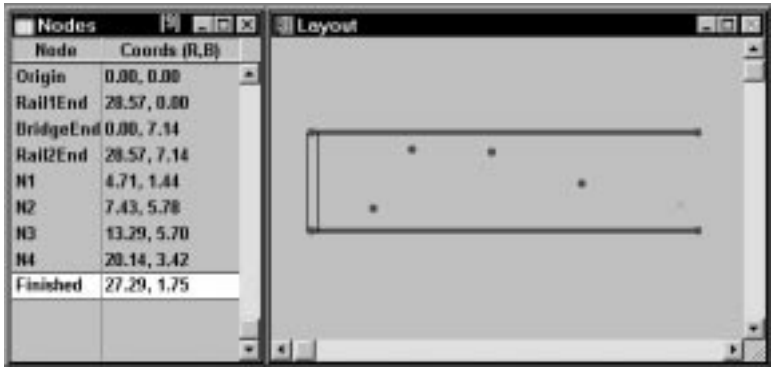
Note Crane coordinates are written as **R,B** where **R** equals the distance from the origin along the axis of the rails and **B** equals the distance from the origin along the bridge axis. Coordinate axis values are initially calculated based on the grid size and scale defined in the General Information dialog.

Movement Nodes

With the crane envelope sized and positioned as desired, you are now ready to place additional nodes inside the envelope. Movement nodes define pickup and drop-off positions, downtime positions, park positions, or home positions (the origin nodes can be used to interface with locations as well). The coordinates of the nodes are calculated based on the envelope coordinate system. You may edit the coordinates of the nodes in the Nodes window. Nodes cannot be placed outside the envelope.

How To — Add nodes in the envelope:

1. Click on the **Nodes** button to display the Nodes window unless the Nodes window is already open.
2. Click inside the crane envelope in the Layout window to add a new node.



3. Edit the node name or coordinates as desired. If the coordinates you enter exceed the boundaries of the envelope, the coordinates will be reset to their previous values when you leave the field. When you change the coordinates of a movement node, the corresponding node graphic in the Layout window is repositioned relative to the rail and bridge coordinates.

How To — How to remove a node from the envelope:

1. Click on the node *record* in the Nodes window.
2. Select **Delete** from the **Edit** menu.

Note

If the non-zero values of the first four pre-defined nodes are changed, all movement node coordinates are recalculated relative to the change in the pre-defined corner nodes, but the envelope graphic remains unchanged. When the envelope's graphic is changed while the Nodes window is open, the user-defined nodes are repositioned relative to their coordinates, but the values of their coordinates do not change.

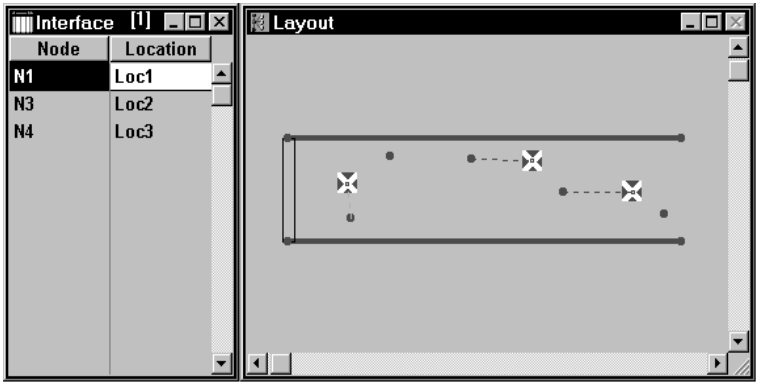
Node-Location Interfaces

You may interface any node including the pre-defined nodes to a location in the layout. If an entity will be picked up or dropped off at a particular location by the crane, that location must be connected to a node in the interfaces table in the Path Networks editor.

How To

Create a node-location interface:

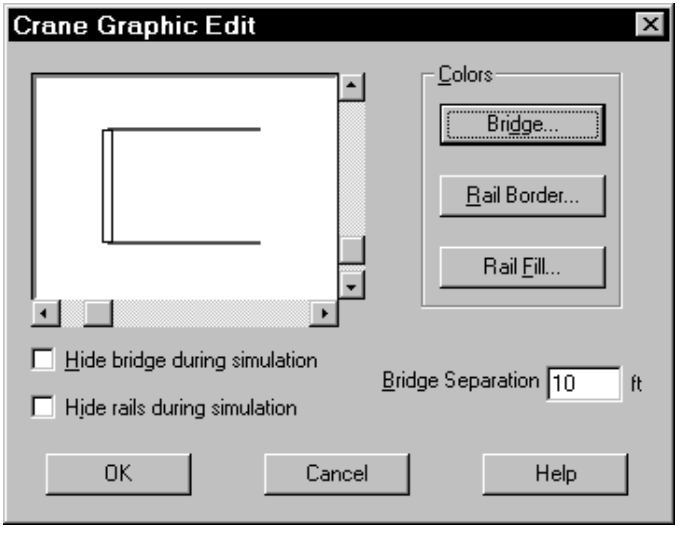
1. Click on the **Interfaces** button in the Path Networks edit table to open the Interfaces window.



2. Left click on the desired node to begin rubber-banding a link or interface.
3. Left click on the desired location to complete the interface.
4. Repeat steps 2 and 3 for each node-location interface, or manually enter node and location names in the Interfaces window. You can also click on the Node and Location buttons in the Interfaces window to select from the lists of available nodes and locations.

Envelope Graphics & Bridge Separation

When you click on the Graphic button in the Path Networks edit table, ProModel displays the Crane Graphic dialog if the selected path network type is a crane. As shown in the following figure, this dialog provides several graphic options including the bridge separation distance.



Scroll Bars Use the scroll bars to change the width of the rails or the bridge.

Colors These three buttons let you define the colors for the crane graphic. The color defined with the **Rail Fill** button is also used for the color of the path nodes in the crane network.

Hide bridge/Hide rails These check boxes allow you to make the bridge and/or rails invisible during simulation.

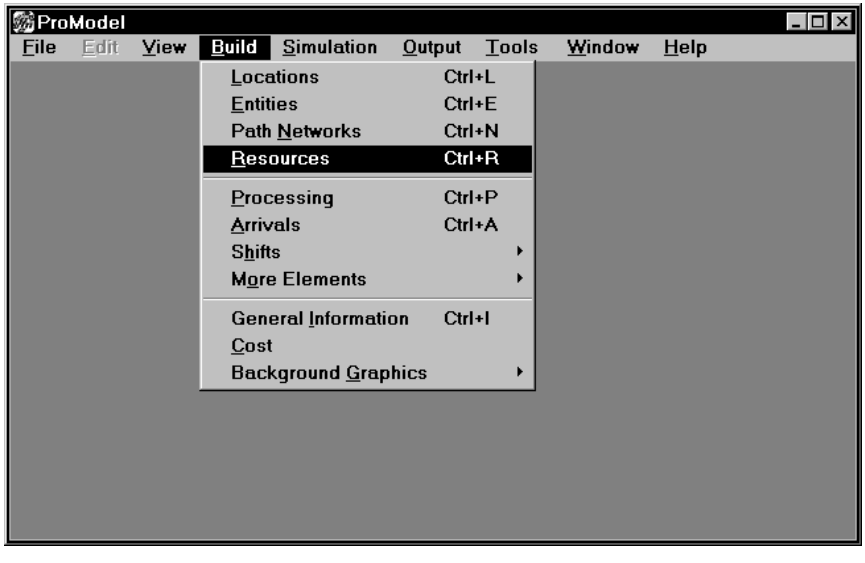
Bridge Separation The minimum distance maintained between multiple cranes operating in the same envelope. This distance is measured from the center-lines of each bridge and is used to ensure that cranes in the same envelope will not crash into each other. The envelope should be extended to allow for any backing away of cranes that could occur.

7.4 Resources

A resource is a person, piece of equipment, or some other device used for one or more of the following functions: transporting entities, assisting in performing operations on entities at locations, performing maintenance on locations, or performing maintenance on other resources. Resources consist of one or more units having common characteristics, such as a pool of service technicians or a fleet of lift trucks. Resources may be dynamic, meaning that they move along a path network, or static, in which no movement occurs. A special type of dynamic resource provided in ProModel is a crane. Resources may also have downtimes.

Every resource has a name and a name-index number. Logic referring to a resource, such as the GET statement, can use either the resource's name, or the RES() function to refer to the resource. The RES() function allows a statement using a resource name to refer to different resources as a simulation progresses. See *Res()* on page 213 of the *ProModel Reference Guide* for more information.

Resources are defined in the Resources Editor, which is accessed through the Build menu.



How To **To create and edit resources:**

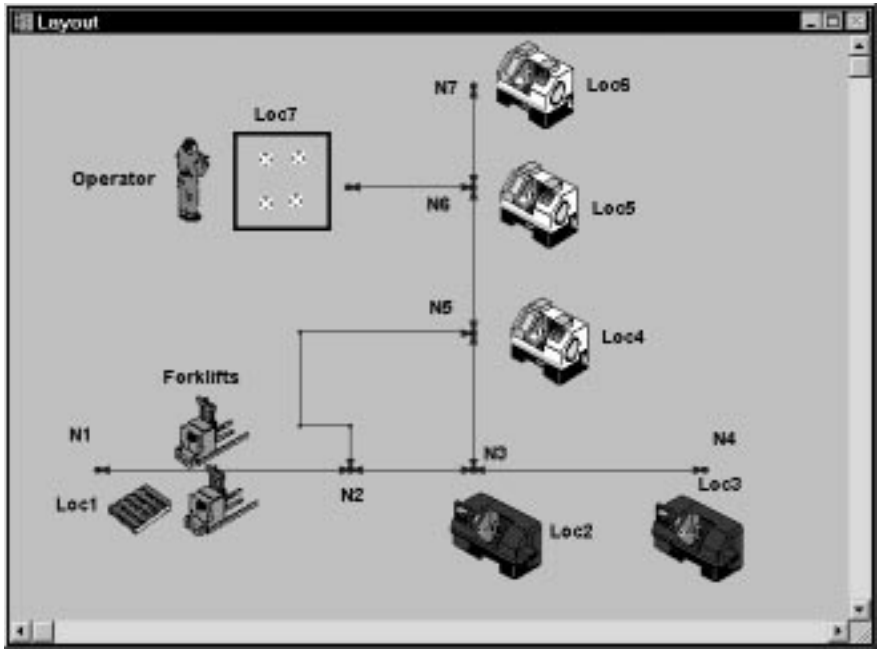
- Select **Resources** from the **Build** menu.

or...

- Right click on the existing location and select **Edit**.

7.4.1 Typical Use of Resources

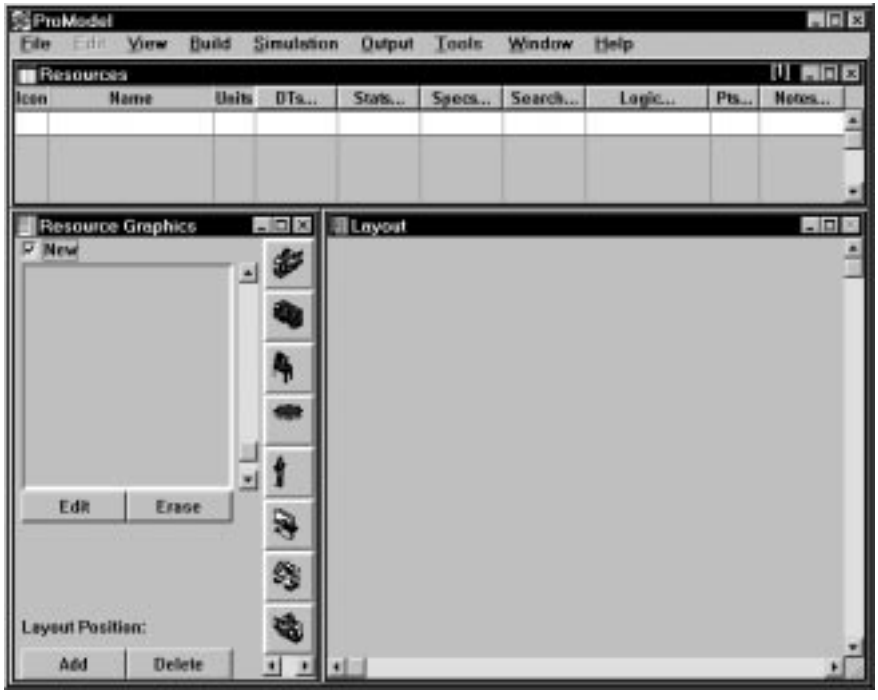
The diagram below shows two types of resources, forklifts and an operator. Forklifts are used as resources to transport entities from Loc1 to any of the processing locations, Loc2 through Loc6. The forklifts are dynamic resources and travel along the path network, Net1, explained previously in *Path Networks* on page 243. The operator inspects all parts at Loc7 and never moves from that location. Therefore, the operator is a static resource and does not need a path network.



The remainder of this section defines the elements and the procedures necessary for specifying static and dynamic resources.

7.4.2 Resources Editor

The Resources Editor consists of the Resources edit table and the Resource Graphics window. These windows are used together to specify the characteristics of a resource.

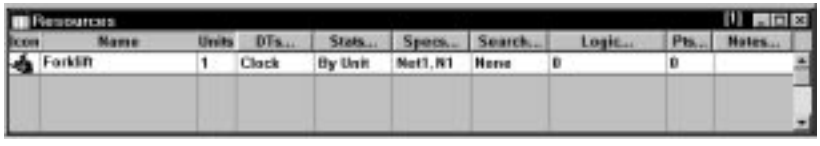


Resources Edit Table Appears along the top of the workspace with fields for specifying the name of each resource, the number of identical units of a resource, the downtime characteristics of each resource, and other important information, such as the path network the resource uses to travel.

Resource Graphics Window Contains graphic icons that may be selected to represent a resource during simulation. A resource may have more than one icon to represent different views of the resource, to change colors when a breakdown occurs, etc. This window also allows you to define multi-unit resources graphically on the layout. Defining a resource is as simple as selecting an icon from the Resource Graphics window, giving the resource a name, and specifying the characteristics of the resource.

Resources Edit Table

The Resources edit table defines the characteristics of each resource in the system. The fields of this table are defined below.



Icon The icon selected for this resource. Icons are selected using the Resource Graphics Window. If more than one icon is selected for the resource, the first icon is shown here.

Name The name of the resource.

Units The number of units represented by this resource name between 0 and 999 (or a macro)—crane resources can have only a single unit. If the entry is a numeric expression, the expression will be evaluated at the start of the simulation run. Consequently, the number of resource units cannot be changed during the simulation run. If you would like to vary the number of units of a resource during runtime, use downtimes to vary the number of resources available at a given time. (See *Resource Downtimes* on page 274.)

Note When you use a macro with a value of zero in the units field, you can use SimRunner to find the optimal number of resources needed for your model.

DTs... Select this field to define any optional downtimes for this resource. Only clock and usage based downtimes are permitted for resources.

Stats... The desired statistics, if any, to gather for this resource. Statistics can be collected as a summary report over all units of a resource, or individually for each unit of a resource. The options are as follows:

- **None** No statistics are gathered.
- **Basic** Average utilization and activity times are recorded collectively for all units of the resource.
- **By Unit** Statistics are gathered for each unit individually as well as collectively.

Specs... Select this field to open the Resource Specifications dialog box, which is used to assign a path network, define the resource speed, pickup and deposit times, etc. For more information on the Specification dialog, see *Resource Specifications Dialog Box on page 282*.

Search... If a path network has been assigned, select this field to access the Work Search and Park Search edit tables, used to define optional work and park searches.

Logic... If a path network has been assigned, select this field to define any optional logic to be executed whenever a resource enters or leaves a particular path node. If you have defined a node entry and exit logic, the logic field will show the number of nodes where node entry and exit logic has been defined.

Pts... If a path network has been assigned, select this field to define resource points, which are auxiliary points where multiple resources may appear graphically when parked or in use at a multi-capacity node.

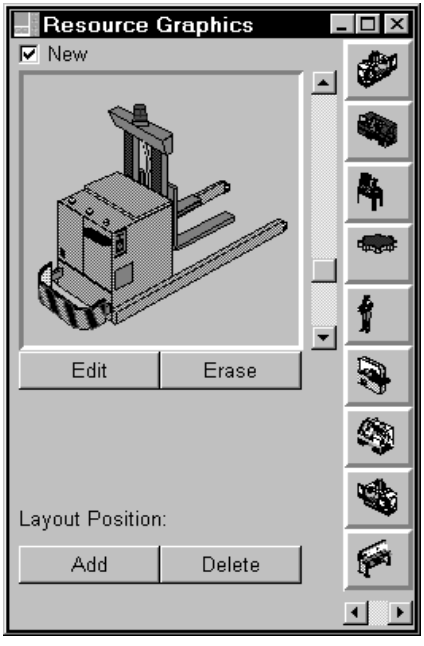
Notes... Enter any notes in this field, or click on the heading button to open a larger Notes window for entering notes.

Resource Graphics Window

The Resource Graphics Window appears when the Resources module is opened and is used to assign graphic symbols to resources. If the New box is checked in the window, selecting a graphic creates a new resource. Multiple graphics are defined for a given resource by selecting the desired resource and unchecking New. This procedure causes a scrollable row of graphic cells to appear which are automatically and sequentially numbered beginning with 1. Graphics may be added or replaced for a given resource by clicking on the desired cell and selecting a library graphic from the graphics menu.

By using the GRAPHIC statement in resource downtime logic, or, in the case of a dynamic resource, node logic, any of the multiple graphics assigned to a resource may be activated during simulation. For static resources, you may define a second or third graphic to be used automatically when the resource is busy or down, respectively.

Resource graphics may be sized using the scroll bar or edited by clicking the edit button. Edit options include rotating, flipping horizontally or vertically, and changing the color of the graphic. In addition, you can also specify the dimensions of the resource graphic. For more information on the Library Graphics Window in the Resource Graphics Window, see *Library Graphics Window* on page 405.



The Layout Position allows you to add or delete resource graphics on the layout. When adding a resource graphic to the layout, ProModel automatically adds a unit and a resource point. When deleting a resource graphic from the layout, ProModel deletes the resource point but leaves the number of units in the units field unchanged.

7.4.3 Static Resources

Static resources are resources that are not assigned to a path network and therefore do not visibly move. A static resource may be needed to perform an operation at only one location, such as an inspection operator, and will appear during the entire simulation in the same place it was defined graphically. Although a static resource is stationary and does not visibly move between locations, it may be used at more than one location or to move entities between locations.

How To

Define a static resource:

1. Select **Resources** from the **Build** menu. This automatically brings up the Resources edit table and the Resource Graphics window, used together to define all resources in the model.
2. Choose a graphic icon for the resource from the Resource Graphics window.
3. Select the **Add** button in the Resource Graphics window.
4. Click on the layout at the desired position of the resource graphic.
5. Add additional resource graphics for the same resource if desired. Every time a resource graphic is placed on the layout for the same resource in the edit table, a new resource point is created.
6. Supply any optional information about the resource, such as downtimes.

Note

Static resources notes:

1. When defining the static resource specifications, the default for Resource Search is Least Utilized. The default for Entity Search is Longest Waiting. You may only specify Pick-up and Deposit Time in the Motion box.
2. There is not a status light for a static resource; however, a second and third graphic may be defined for use when the resource is busy or down, respectively. If no second and third graphic are defined, the resource graphic changes color to green when in use and red when down.

7.4.4 Dynamic Resources

Dynamic resources are resources that move along an assigned path network and may transport entities between locations as a forklift would. They may also need to process entities at several locations, such as an operator performing tasks at more than one location. For these reasons, it is usually preferable to model the resource's movement using a path network. Defined properly, the resource will travel along the path network during the simulation run.

How To

Create a dynamic resource:

1. Create a path network using the Path Network Editor.
2. Select **Resources** from the **Build** menu. This automatically brings up the Resources edit table and the Resource Graphics window, used together to define all resources in the model.
3. Choose a graphic icon for the resource from the Resource Graphics window.
4. Click the **Specs...** button to open the Specifications Dialog.
5. Assign a path network to the resource.
6. If desired, place units of the resource on the layout by selecting the Add button in the Resource Graphics window and clicking on the layout. Every time you create and place a resource graphic on the layout for the same resource in the edit table, ProModel creates a new resource point. See *Resource Points* on page 290 for more information.
7. Supply any optional information about the resource including number of units, downtimes, work and/or park searches, and node logic in the Resources edit table.

Note

Dynamic resources notes:

1. When defining the resource specifications, the default Resource Search for dynamic resources is Closest Resource. The default for Entity Search is Closest Entity.
2. More than one resource can use the same path network.

7.4.5 Crane Resources

With the crane envelope and node-location interfaces defined in the Path Networks module, you are ready to define one or more cranes to operate within the envelope. Cranes are defined in the Resources module where you specify the hoist graphic, speeds, downtimes, priorities, and logic.

How To **Define a crane resource:**

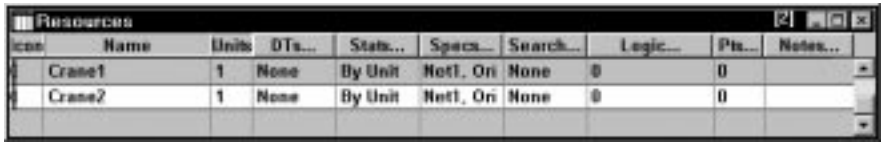
1. Open the Resources module from the Build menu.
2. Select a graphic for the hoist from the Resource Graphics window.
3. Enter a name for your crane in the Resources edit table.
4. Click on the Specs button in the edit table and set the path network to one of the previously defined crane networks. Set the other specifications.

Crane Specifications

The following paragraphs cover the specifications and other important considerations to take into account when defining a crane.

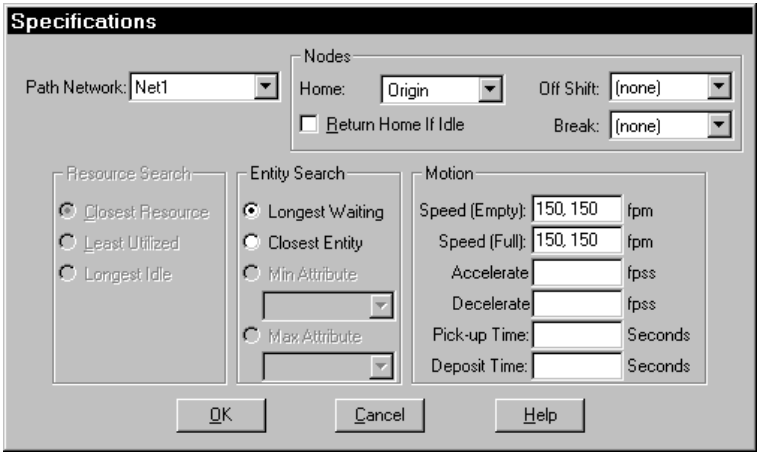
Crane Graphics In the Resources module, the selected graphic represents the hoist. Multiple graphics for the same hoist may be defined, edited and sized in the same way you create multiple graphics for other resources.

Units & Multiple Cranes Crane resources are limited to one unit. Multiple cranes operating in the same envelope are defined as separate resources, each assigned to the same crane network. See *Crane Operations* on page 7 of the *ProModel Reference Guide* for more information on multiple cranes.



Downtimes Clock and Usage downtimes may be defined for cranes. In addition to these downtimes, you can assign cranes to shifts in the same way other resources are assigned to shifts. When going down, cranes move to the downtime node specified in the DTs dialog. If there are no nodes associated with a downtime, the crane is taken down at its current position.

Specifications Separate values must be entered for bridge and hoist motion parameters. **Speed (Empty)**, **Speed (Full)**, **Accelerate** and **Decelerate** values must be entered for bridge movement along the rails and hoist movement along the bridge. In each field, the format is *Bridge Value, Hoist Value*. Speed values must be entered. However, acceleration and deceleration values are optional; if they are left blank, infinite acceleration and deceleration are assumed (**Hint:** this provides better run-time computational efficiency). Also note that acceleration and deceleration values are always entered in *feet per second per second* (fpss)



Crane Searches Define Work and Park searches in the same way you define searches for other resources. For more information, see *Nodes, Work, and Park Searches* on page 15 of the *ProModel Reference Guide*.

Node Logic Entry and exit logic for cranes at nodes may be defined in the same way node logic is defined for other resources.

Using the Crane & Crane Priorities

Enter statements defining crane usage in the Processing module in the operation or move logic windows.

Cranes may be captured by priority just like any other resource. In addition, you may also set move priorities so cranes assigned to the same envelope competing for the same zone may have priority over each other. In ProModel, the GET, JOINTLY GET, and USE statements have an additional priority field to make the move priority assignment. For example, **GET CRANE1, 100, 200** sets a capture priority of 100 and a movement priority of 200. When the requestor is competing with other requestors to capture the crane, it has a priority of 100. Once the crane is captured, it has a movement priority of 200 to compete with cranes in the same envelope.

In move logic, you may define up to three priorities for cranes when using the MOVE WITH statement: capturing the crane, moving the crane to pickup the entity, and moving the crane to deliver the entity. For example, the following statement moves the entity with Crane1 with the following priorities set: a capture priority of 100, a pick up priority of 200, and a delivery priority of 300.

MOVE WITH Crane1, 100, 200, 300

The default move priority for a crane is zero. In fact, all crane priorities are reset to zero after each move. For more information see *Crane Priorities, Preemption & Bridge Bump-Away* on page 6 the *ProModel Reference Guide*.

Animation & Run-time Features

During a simulation run, entities picked up by a crane appear graphically on the entity spot of the hoist. The hoist graphic appears above the entity graphic, and the bridge of the crane appears on top of the entity and hoist graphics.

7.4.6 Multiple Resource Graphics

ProModel allows you to define several different graphic icons for the same resource. For example, you may wish to change the color of a resource whenever it becomes unavailable due to an unscheduled downtime. Resource graphics for dynamic resources may be changed during a simulation by using the GRAPHIC statement (see *Graphic* on page 160 of the *ProModel Reference Guide*) in either the node or downtime logic. The GRAPHIC statement for static resources can only be used in downtime logic, however, any second and third graphics are automatically used when static resources are busy or down, respectively. If no second and third graphics are defined, the resource graphic turns green when in use and red when down.

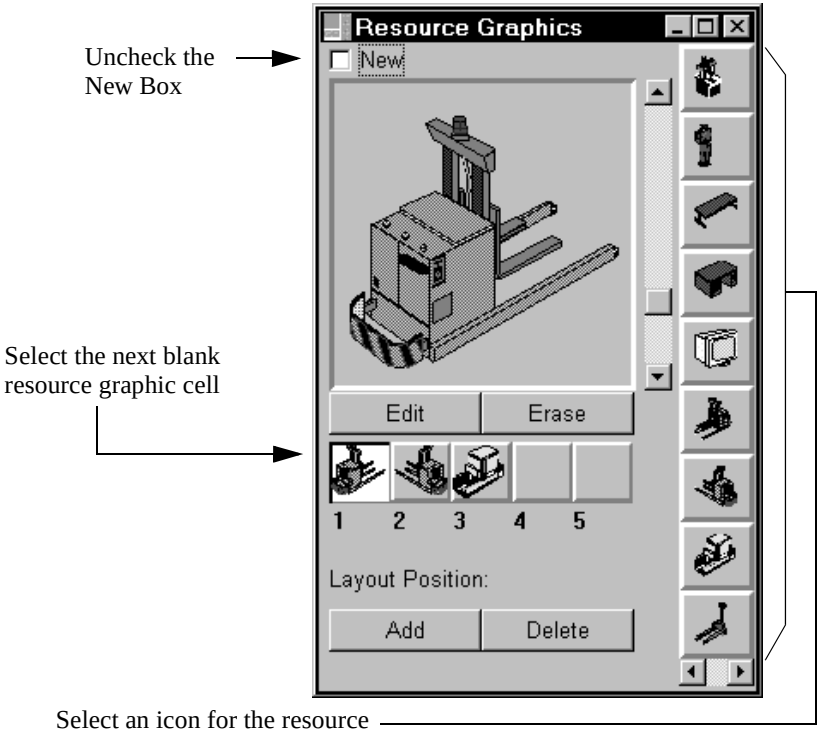
How To

Assign multiple graphics to a resource:

1. Select **Resources** from the **Build** menu.
2. Highlight the desired resource in the Resources edit table.
3. *Uncheck* the New box in the Resource Graphics window.
4. Click on the next blank resource graphic cell in the Resource Graphics window.
5. Select the desired resource icon.
6. Change the color, rotation, etc. of the new graphic by clicking on the Edit button.

An example of a single resource with multiple graphics is given on the following page.

Multiple Resource Graphics Example



This example shows a forklift with two opposite orientations. You may define as many graphics as needed for each resource.

Note When changing the graphic for a resource in downtime logic, or in the case of dynamic resources, node logic, the number after the word GRAPHIC refers to the cell number in the Resource Graphics window as shown previously. For example the statement GRAPHIC 2 would change the forklift to the icon in cell number 2. The default graphic is the graphic in cell number 1 if none is specified. See *Graphic* on page 160 of the *ProModel Reference Guide* for more information.

7.4.7 Multi-Unit Resources vs. Multiple Single-Unit Resources

Multi-Unit Resources

When a resource is defined as having more than one unit, each resource unit is given a numeric suffix by which it is identified in the output report. For example, a resource (Res1) which has five units will display output statistics for resources called Res1.1, Res1.2, ... Individual units of a resource (e.g., GET Res1.1) cannot be requested.

Suppose you define a resource, Operator, which has ten units. You also have ten locations and each resource unit can interface only with one location. For example, Res1.5 can work only at Loc5. Since "USE Res1.5" is invalid, you would need to use multiple single-unit resources instead.

Multi-unit resources are intended for use when several resources are defined with the exact same specifications and any resource can be used at a number of locations. For example, a lathe can be operated by one of three operators. If you did not use multi-unit resources, you would need to specify "USE Res1 OR Res2 OR Res3," although this can easily be abbreviated by using a macro to represent the resource expression. When you define three units for a single resource, Res1, you can simply state "USE Res1" and one resource unit will be used based on its availability.

Multiple Single-Unit Resources

Multiple single-unit resources are useful when resources have different specifications, follow different path networks, or are used at specific locations. If several resources have the same specifications and travel the same path network but can only do work or interface with specific locations, they must be defined as multiple single-unit resources. This is because a unit of a multi-unit resource must be able to interface with all locations where it is called to work.

See also *Resource Grouping* on page 446.

7.4.8 Resource Downtimes

Resource downtimes refer to the times when a resource is unavailable due to scheduled events like breaks and shift changes, or unscheduled events like illness and random failures. For scheduled events, it is much easier and more straightforward to define these downtimes using the shift editor (see *Shifts & Breaks* on page 323). Unscheduled downtimes based on the elapsed time of the simulation clock or resource usage time are defined in the Resources edit table by clicking on the downtime heading button.

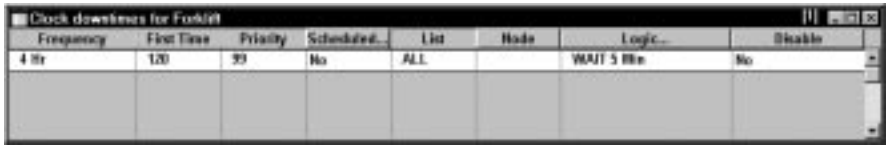
How To **Define resource downtimes:**

1. Select **Resources** from the **Build** menu.
2. Select the resource for which the downtime is to be defined.
3. Click the **DTs...** button from the Resources edit table.
4. Select the downtime basis: **Clock** or **Usage**.
5. Enter the required information in either the Clock Downtime or Usage Downtime edit table. Each of these tables is described in the following pages.

Note Unlike location downtimes, multiple resource downtimes occurring within the same time frame are processed sequentially, not concurrently. However, through the use of the DTDelay function, concurrent downtimes can be achieved for resources.

Clock-Based Downtime

Clock-based downtimes for resources are specified through the Clock Downtimes edit table shown below. The fields of this table are defined as follows:



Frequency The time between downtimes. This may be a constant time as shown above, a distribution, or an expression.

First Time The time of the first downtime occurrence. Leave this field blank if the first occurrence is to be determined from the frequency field.

Priority The priority of the downtime (0-999). The default priority is 99, which is the highest non-preemptive priority.

Scheduled... Select YES if the downtime is to be counted as a scheduled downtime. Select NO if the downtime is to be counted as a non-scheduled downtime. (All scheduled downtimes are deducted from the total hours scheduled in the statistical calculations.)

List A list of the individual units of the resource to be affected by the downtime. You may list individual units of the resource, specify ALL, or leave blank to affect all units.

- **1,2** Units 1 and 2 only
- **1-3,5** Units 1 through 3 and 5 only
- **none** You may use none to indicate that no unit will adopt this downtime. This is useful in creating a run-time interface. By using a macro to represent the number of units, the user may select none as an option.
- **Macro** The name of a run-time interface macro that allows the user to define the units to be affected by the downtime.

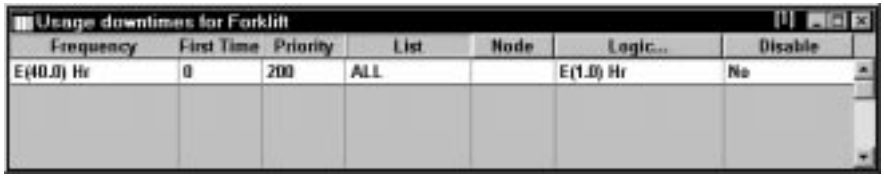
Node This field applies only to dynamic resources and defines the node to which the resource will travel to go down. If no node is entered, the resource stays at the current node. The actual downtime will not begin until the resource arrives at this node. Traveling to the downtime node is counted statistically as time traveling to park.

Logic... Specific logic to be performed when the downtime begins, typically a WAIT statement. Resources may be used to service resources that are down if the servicing resource is static, or if the servicing resource is dynamic and uses the same network. (See *Appendix A Expressions & Statements* on page 259 of the *ProModel Reference Guide* for a list of statements valid in Resource Downtime logic.)

Disable Select YES to disable a downtime without removing it from the table.

Usage-Based Downtime

A usage-based downtime is a downtime based on how long a resources has been used, such as how often a forklift needs to be refueled. Usage-based downtimes for resources are specified through the Usage Downtimes edit table shown below. Actual time in use includes any time that a resource is moving with an entity or is being used by an entity at a location. It also includes any time a resource is being used in downtime logic as a maintenance resource. The fields of this table are defined as follows:



Frequency The time between downtimes, based on the usage time of a resource. This may be a time distribution as shown above, or an expression.

First Time The time of the first downtime occurrence. Leave this field blank if the first occurrence is to be determined from the frequency field.

Priority The priority of the downtime (0-999). The default priority is 99, the highest non-preemptive priority.

List A list of the individual units of the resource to be affected by the downtime. You may list individual units of the resource, specify ALL, or leave blank to affect all units.

- **1,2** Units 1 and 2 only
- **1-3,5** Units 1 through 3 and 5 only
- **none** You may use none to indicate that no unit will adopt this downtime. This is useful in creating a run-time interface. By using a macro to represent the number of units, the user may select none as an option.
- **Macro** The name of a run-time interface macro that allows the user to define the units to be affected by the downtime.

Node This field applies only to dynamic resources and defines the node to which the resource will travel to go down. If no node is entered the resource stays at the current node. The actual downtime does not begin until the resource arrives at this node. Traveling to the downtime node is counted statistically as time traveling to park.

Logic... Specific logic to be performed when the downtime begins, typically a WAIT statement. Resources may be used to service resources that are down if the servicing resource is static, or if the servicing resource is dynamic and uses the same network. For a list of statements valid in Resource downtime logic, see *Appendix A Expressions & Statements* on page 259 of the *ProModel Reference Guide*.

Disable Select YES to disable a downtime without removing it from the table.

Note Usage-based downtimes do not accumulate. For example, if a downtime is preempted by an entity and another downtime is scheduled to occur while processing the entity, only the first downtime resumes after processing the entity. All others are ignored.

7.4.9 Resource Priorities and Preemption

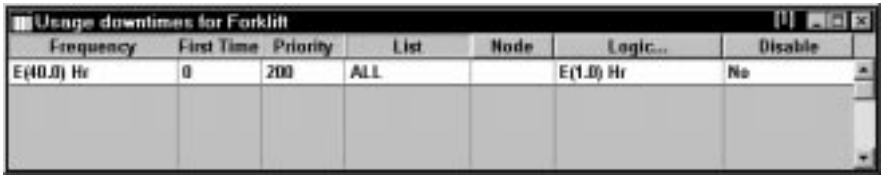
Priorities for resource requests may be assigned through a GET, JOINTLY GET, or USE statement in operation logic, downtime logic, or move logic (or the subroutines called from these logics). Priorities for resource downtimes are assigned in the Priority field of the Clock and Usage downtime edit tables. The following examples illustrate these points.

Process Table

Entity	Location	Operation (min)
EntA	Loc1	Use Res 1,200 For N(3,.1)

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	First 1	MOVE FOR 5



When an entity using a resource is preempted by either a downtime or another entity, any processing time for the preempted entity due to a WAIT or USE statement is interrupted until the preempting entity or downtime releases the resource. If an entity is using other resources in addition to the one preempted, the other resources remain in possession of the entity.

In the case of a resource downtime preempting another resource downtime, any remaining time delay, as well as any other downtime logic remaining to be processed by the preempted downtime, is immediately discontinued without resuming and the preempting downtime takes over.

Note

Resource priorities and preemption notes:

1. If a resource is transporting an entity, it cannot be preempted by another entity or by a downtime until it drops off the current entity at the destination location. Therefore, the resource will deliver the current entity and then immediately come under control of the preempting entity or downtime.
2. If a resource is moving but does not possess an entity, the resource can be preempted by a downtime or entity. The resource will stop at the next node in the path network and travel to the downtime node after which the resource will go down. A crane resource will try to stop at its current position unless there are other bridges in the same envelope with conflicting claims of greater than or equal move priorities.

7.4.10 Resource Shift Downtime Priorities

In ProModel, you define the shift downtime priorities in the Shift Assignments module. The priority for a resource to start a shift downtime and the priority required for some other task to preempt the downtime must be set in the Shift Assignments module.

Although a resource may be in use during a shift downtime, the scheduled hours in the statistics will still reflect the hours scheduled to be on shift. For example, a resource goes off shift after eight hours. Due to an emergency, the resource is called back two hours later to work on a machine that has gone down. The statistics will still indicate that the scheduled hours for the resource are eight when the resource actually spent more than eight hours in use, because the resource was scheduled to work only eight hours. The resource's total usage time, however, will indicate the additional time spent working on the downed machine.

7.4.11 Resource Preemption Matrix

The following Preemption Matrix shows the possibilities of entities and downtimes preempting each other in the use of a resource. “Current” refers to the entity or downtime in possession of the resource when the requesting entity or downtime attempts to capture it. Downtimes below refer to clock and usage-based downtimes only.

Priority values are divided into ten levels (0 to 99, 100 to 199, ..., 900-999), with values beyond 99 used for preempting entities or downtimes of a lower priority level.

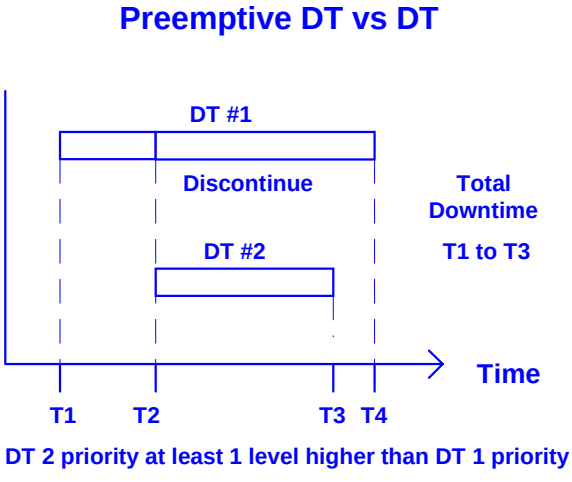
	To Preempt The Current Owner	**To Preempt The Current Downtime**
Requesting Entity or Another Resource or Location’s Downtime	1 priority level higher	2 priority levels higher
Requesting Downtime	1 priority level higher	1 priority level higher

- The upper left quadrant shows that if an entity tries to seize a resource currently owned by another entity (or another resource’s downtime or a location downtime), the entity must have a priority at least one level higher than the current entity to preempt the resource.
- The lower left quadrant shows that a downtime must have a priority at least one level higher than the entity currently owning a resource if the resource is to be preempted.

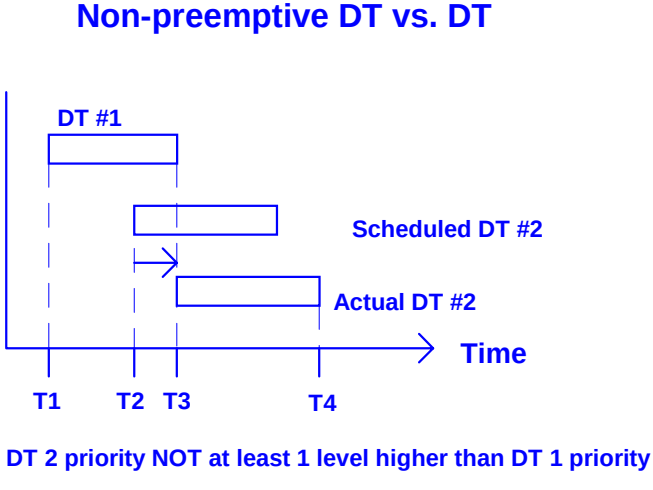
- The upper right quadrant shows that an entity must have a priority at least 2 levels higher than the current downtime priority in order to preempt a downed resource.
- The lower right quadrant shows that a preempting downtime must have a priority at least one level higher than the current downtime to preempt it.

The following graphics demonstrate the previous explanations.

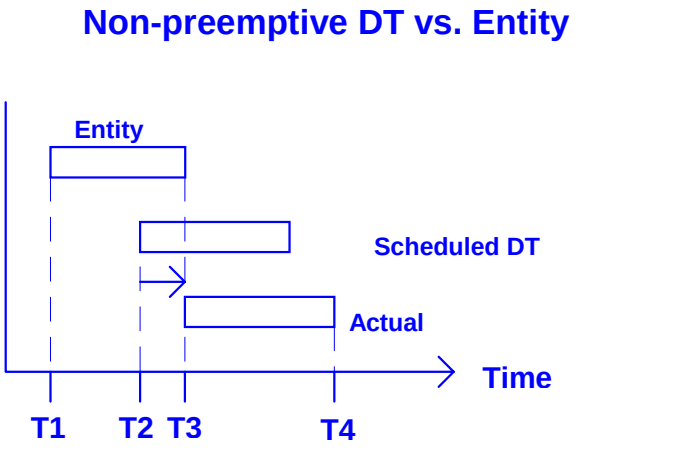
Example 1



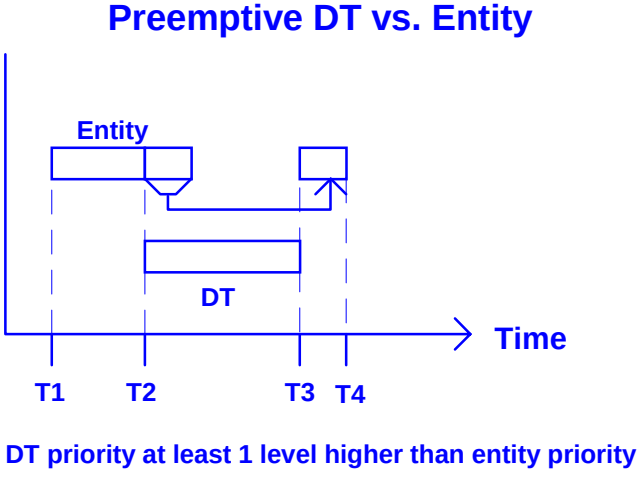
Example 2



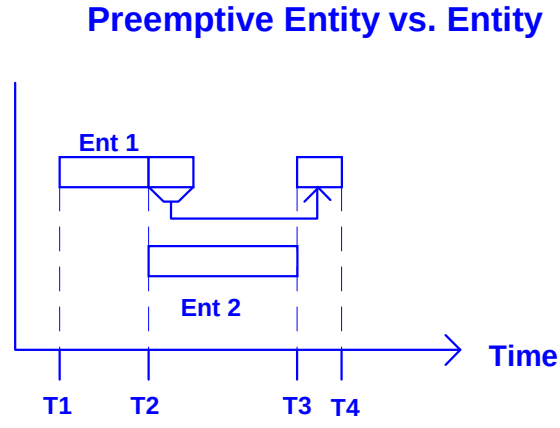
Example 3



Example 4

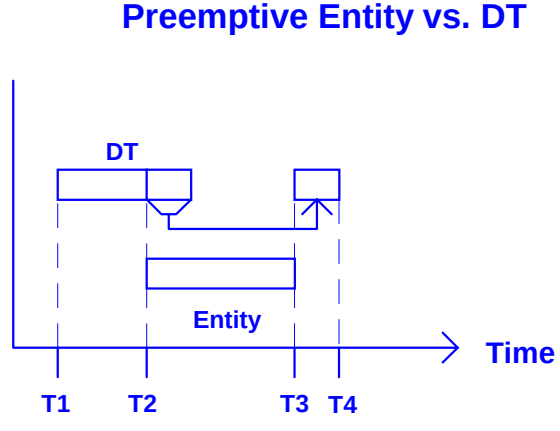


Example 5



Ent 2 priority at least 1 level higher than Ent 1 priority

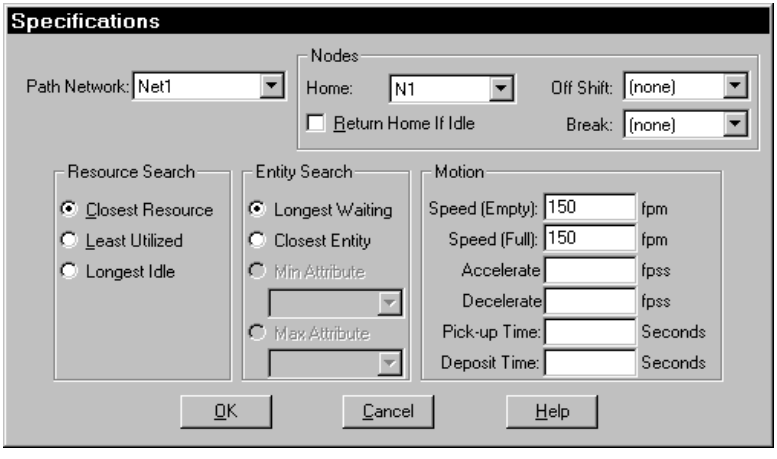
Example 6



Entity priority at least 2 levels higher than DT priority

7.4.12 Resource Specifications Dialog Box

The Specifications dialog box contains information for defining the operating characteristics of each resource in the system. Many of the items pertain only to dynamic resources (i.e., resources that have path networks). If the resource is static (i.e., it is not assigned to a path network) many of the options will be disabled. The Resource Specifications dialog box includes the following fields:



Path Network The name of the path network along which the resource travels. This should be "none" for a static resource.

Home If a path network has been assigned, this is the name of the home node, which is the node where the resource is positioned at the beginning of the simulation.

Note To have resources start at other nodes in the network, define a Park Search from the home node which causes the resource unit or units to be positioned at the nodes identified in the park search.

Return Home if Idle If a path network has been assigned, check this box to return the resource to the home node when no other tasks are waiting to be performed and no park searches are defined.

Off Shift If the resource is assigned to a path network and shift, this is the node to which the resource travels to go off shift.

Break If the resource is assigned to a path network and shift, this is the node to which the resource travels to go on break.

Resource Search When an entity that needs a resource must select between several available resource units, it follows this rule. This only applies to multi-unit resources. The following rules may be specified:

- **Closest Resource**
- **Least Utilized Resource**
- **Longest Idle Resource**

Note For a non-passing path network, only the *Closest Resource* rule is allowed since the other rules could cause the network to jam. Static resources only allow *Least Utilized* and *Longest Idle* rules since there is no traveling to be done.

Entity Search When two or more entities of equal priority request a resource at the same time, the resource follows this rule to choose the one to service. The resource first checks for any entities waiting at locations listed in a work search before defaulting to this rule, and if an *exclusive* work search has been defined, the default entity search rule is not used. The following rules may be specified:

- **Longest waiting entity** (with highest priority)
- **Closest entity** (with highest priority)
- **Entity with the minimum value of a specified attribute**
- **Entity with the maximum value of a specified attribute**

Note Entities look for resources to move them after they capture a location. If several entities are waiting to be transported to one location by a resource, and you want the entity with the minimum attribute value to arrive next at the location, you must use the Locations edit table to define the rule at the location for incoming entities as minimum attribute value. The Closest entity rule applies to dynamic resources only.

If the path network that the dynamic resource is using is time-based, the closest entity is the entity with the least number of nodes from the resource. If the path network is defined by speed and distance, the closest entity is the entity with the shortest distance from the resource.

Motion If a path network has been assigned, the motion fields define the speeds and times required for basic resource movement and contain the following information:

- **Speed traveling empty/full***
- **Acceleration rate***
- **Deceleration rate***
- **Pickup time**

- **Deposit time**

  *For crane resources enter two values: **bridge, hoist**.

Note

Resource specification notes:

1. The units to the side of these fields change automatically from feet to meters (and vice-versa) depending on the default distance units selected in the General Information dialog box.
2. Pickup and deposit times for resources are included as transit time in the output statistics.

7.4.13 Resource Search Routines

Search routines refer to the instructions a dynamic resource will follow after being freed at a path node where a search routine was defined. Two types of search routines may be specified.

Work Search A list of *locations* where entities may be waiting for the resource. Work searches may be either *exclusive* or *non-exclusive*.

- Use exclusive work searches to limit the locations a resource may search for work. An exclusive work search will cause the resource to search *only* the locations in the list. If no work is found at any of the listed locations, the resource will either park at a node listed in its park search, go to its home node, or simply become idle until work is available at one of the locations in the exclusive work search list.
- Use non-exclusive work searches to have a resource check for work at certain locations first, and then move on to others. A non-exclusive work search will cause the resource to search the listed locations first and then resort to the default search rule listed in the Resource Search section of the Specifications dialog box (e.g., oldest waiting entity, closest waiting entity).

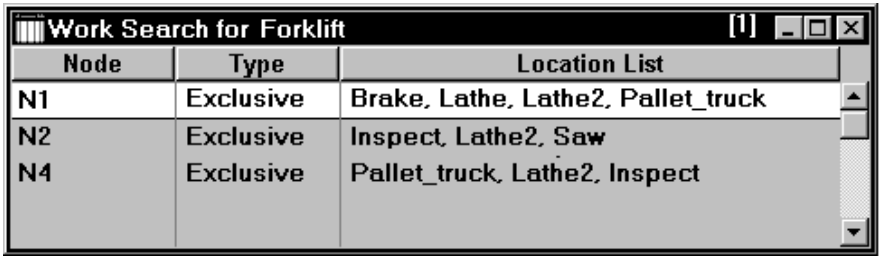
Park Search Park searches are typically used to get a resource off a main path segment, or send the resource to the next most likely place work will become available. A park search is a list of *nodes* to which a resource may be sent to park if no work is waiting at either the work or default search locations.

Note

A lockup can occur in the model if you define a park search at the home node and specify **Return Home If Idle** in the Resource Specifications.

Work Search Edit Table

Work searches are defined for dynamic resources through the Work Search edit table shown below. If several resources share the same path network, each resource must have its own work search defined (i.e., resources cannot share work searches).



The fields of the Work Search edit table are defined as follows:

Node The node for which the work search is defined.

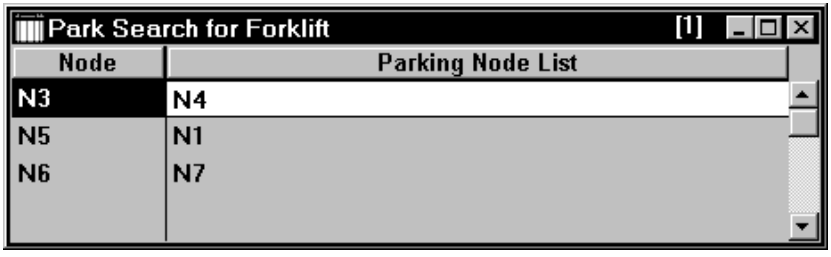
Type Exclusive or non-exclusive work search. See *Resource Search Routines* on page 284 for an explanation of *exclusive* and *non-exclusive* work searches.

Location List A list of *locations* to be searched for waiting entities when the resource becomes available.

In the edit table above, work searches have been defined at each location where the forklift delivers entities. The purpose of the work search in this example is to force the forklifts to give priority to entities waiting to be delivered to processing locations before taking an entity to a non-processing location. This helps to keep the processing locations busy at all times.

Park Search Edit Table

Park searches are defined for dynamic resources through the Park Search edit table shown next. If several resources share the same path network, each resource must have its own park search defined (i.e., resources cannot share park searches).



The fields of the Park Search edit table are defined as follows:

Node The node for which the park search is defined.

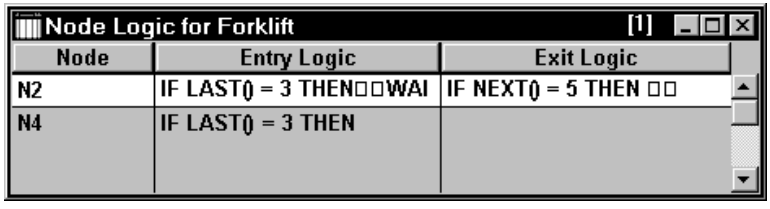
Parking Node List A list of *nodes* to which a resource may be sent to park. The resource will look for available capacity at the first node in the list (Node capacity is defined in the Path Networks edit table in the Nodes window). If there is no capacity at that node, the resource will look to the second node in the list and so on until a node with capacity is found. If no capacity is found at any node in the list, the resource will remain where it is until capacity becomes available at one of the nodes in the list.

In the table above, a park search has been defined for each of the internal path nodes where a forklift might deliver an entity and then be in the way if it has nothing else to do. Specifically, if a forklift makes a delivery and then becomes available at node N3, it will park at node N4. From node N5 it will park at node N1, and from node N6 it will park at node N7.

7.4.14 Node Logic Editor

The Node Logic edit table is used to define special logic for a dynamic resource to perform upon entering or exiting a node. Node logic may be defined for any dynamic resource at any node. Typical uses of node logic are:

- Changing a resource graphic using the GRAPHIC statement
- Controlling traffic using WAIT UNTIL statements
- Gathering special statistics on resource movement



The fields of the Node Logic edit table are defined as follows:

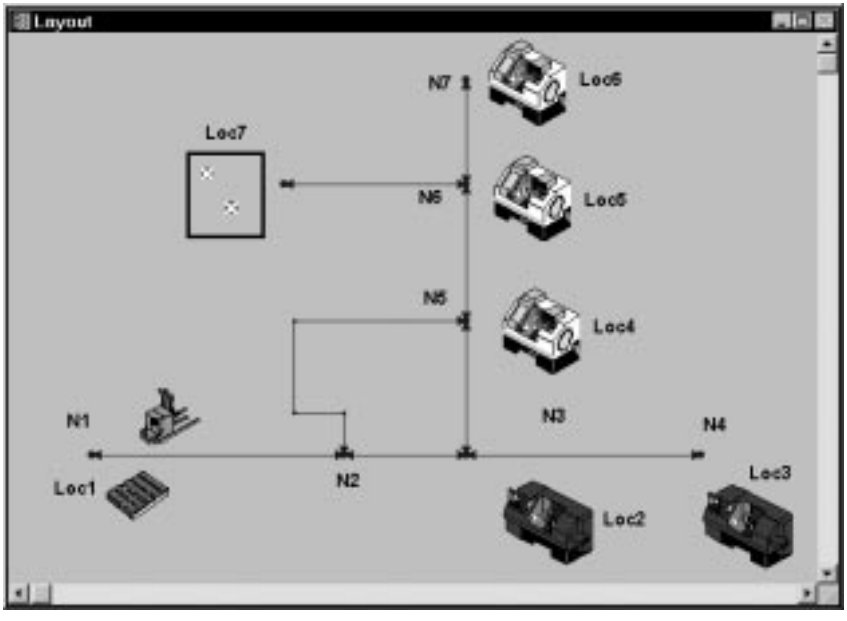
Node The entry or departure node where the resource will process the logic.

Entry Logic Logic to be executed when a resource enters the node.

Exit Logic Logic to be executed when a resource leaves the node.

The table above was taken from the example contained on the following pages.

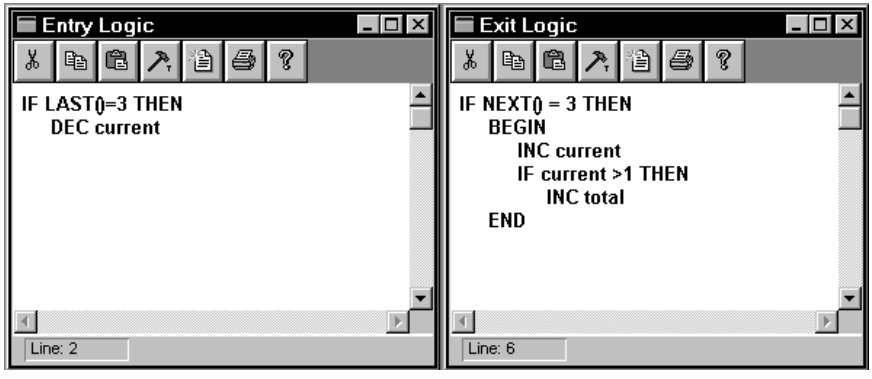
Node Logic Example



Suppose that for safety considerations we desire to keep track of the number of times that both forklifts simultaneously enter a particular zone of the network consisting of branches N2 to N3, N3 to N4, and N3 to N5. (It is thought that this zone may be particularly susceptible to accidents due to heavy traffic.) We could accomplish this using node logic at the entry and exit points of the zone.

The only way to enter or exit the zone is through nodes N2 and N5. To track the number of forklifts currently in the zone, we increment and decrement a variable called Current. Each time a forklift *leaves* node N2 or N5 en-route *to* node N3 we increment variable Current. Each time a forklift *enters* node N2 or N5 enroute *from* N3 we decrement variable Current. Finally, each time we increment the variable Current, we check to see if Current > 1. If so, we increment a second variable called Total to record an occurrence of both forklifts in the zone at the same time.

The following windows show the entry and exit logic for node N2, representing one entry to the zone. The node logic for node N5 is identical to that for node N2.

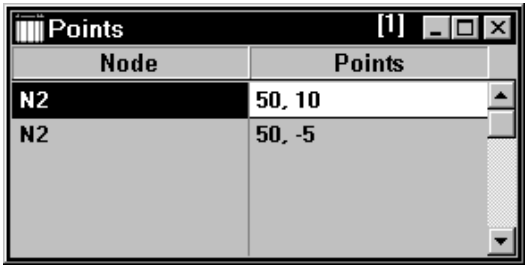


Note This example follows the rule that allows the LAST() function to be used only in Node Entry Logic, while the NEXT() function may be used only in Node Exit Logic. (See *Functions* on page 81 of the *ProModel Reference Guide* for details.)

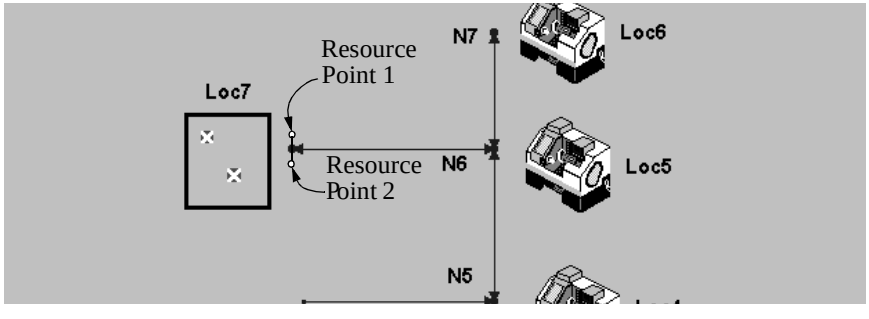
7.4.15 Resource Points

For a static resource, resource points are the layout coordinates of the resource graphics. For dynamic resources, resource points are auxiliary points where multiple resources may appear graphically when in use or parked at a multi-capacity node. When a crane resource arrives at a node to perform a task, the hoist graphic appears on the resource point if one is defined (this can be used to achieve a three dimensional effect). When a resource arrives at a node, it will appear on that node unless a resource point is defined for that resource at that node. The resource will appear on the resource point (except for cranes as mentioned above) when it arrives to park or perform a task at a particular node. Resource points prevent resources from appearing on top of each other. In the case of dynamic resources, resource points are defined in terms of an offset from the node to which they are connected. Resource points are defined in terms of an offset from the upper left corner of the layout for static resources.

The following Resource Points edit table shows that node N2 has two resource points attached to it. The horizontal offset is 0 units for each point. The vertical offset is 15 units both up and down from the node position. (For resource points positive distances are up and to the right.)



Whenever a forklift arrives or parks at node N2, it will automatically go to one of the two resource points. This prevents the forklifts from graphically appearing on top of each other if they are both at node N2 simultaneously.



How To **Add resource points to a node:**

1. Select **Resources** from the **Build** menu.
2. Click on the resource points heading button, **Pts....**
3. Click on the node for which a resource point is to be added.
4. Click on the layout where the resource point is to appear.
5. Repeat steps 3 and 4 for each resource point to be added.

Note

Adding resource points notes:

1. Resource points are automatically added to the home node for each resource graphic placed on the layout.
2. Defining multiple resource points at the same node is redundant for cranes since crane resources can only have a single unit.

How To **Delete resource points:**

1. Bring up the Resource Points edit table by clicking on the **Pts...** heading button in the Resources edit table.
2. Select the point to be deleted by highlighting the edit table record or clicking on the resource point in the layout. (If you click on the point to select it, you must then reactivate the Resource Points window by clicking on the title bar of the table.)
3. Select **Delete** from the **Edit** menu.

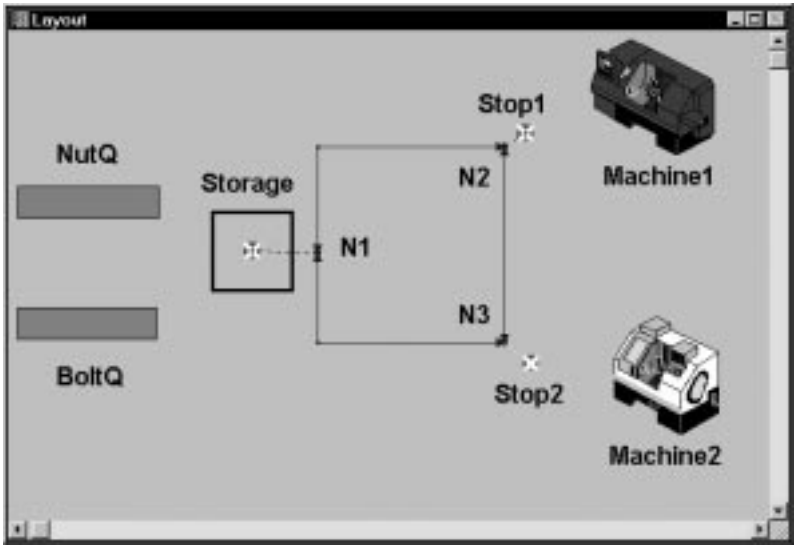
How To **Move resource points:**

1. Bring up the Resource Points edit table by clicking on the **Pts...** heading button from the Resources edit table.
2. Drag the resource point to a new location with the mouse.

7.4.16 Resource Example

In some applications, it is necessary to transport a batch of parts with a resource. However, ProModel allows a resource to only have a capacity of one. To work around this, you can group the parts together and transport the grouped entity with a resource. The following model demonstrates how to transport a batch of parts with a resource.

Suppose an Automatic Guided Vehicle (AGV) picks up, transports, and delivers boxes of nuts and bolts to different locations. The nuts and bolts initially arrive at queues, Nut_Q and Bolt_Q, The boxes of nuts and bolts are loaded onto a pallet. A resource, AGV, transports the pallet with boxes of nuts and bolts to its destination. The nuts must be delivered to Stop1 after which they are processed at a machine, Mach_1. The bolts must be delivered to Stop2 after which they are processed at a machine, Mach_2. After the pallet does not contain any more boxes, the AGV transports the pallet back to the original location, Storage.



In the system being modeled, the AGV can carry up to three boxes. If all three boxes are bolts, the AGV will pass Stop1 and stop and unload the bolts at Stop2. Likewise, if all three parts are nuts, the AGV will stop at Stop1 and unload the nuts but will pass Stop2.

For modeling purposes, we defined four entities:

Entity	Description
NutBox	A box of nuts
BoltBox	A box of bolts
Pallet	The entity on which the boxes of nuts and bolts are loaded
Box	A combined entity consisting of boxes of nuts and bolts

Consider the following processing logic:

Process Table

Entity	Location	Operation (min)
NutBox	Nut_Q	
BoltBox	Bolt_Q	
Pallet	Storage	LOAD 3 IF GROUPQTY(NutBox)>0 THEN ROUTE 1 ELSE ROUTE 2
Box	Stop1	UNLOAD 3 IFF ENTITY()=NutBox IF GROUPQTY(BoltBox)=0 THEN ROUTE 1 ELSE ROUTE 2
NutBox	Stop1	
NutBox	Mach_1	WAIT 5
Box	Stop2	UNLOAD 3 IFF ENTITY()=BoltBox
BoltBox	Stop2	
BoltBox	Mach_2	WAIT 3

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	NutBox	Storage	LOAD 1	
1	BoltBox	Storage	LOAD 1	
1 2	Box Box	Stop1 Stop2	FIRST 1 FIRST 1	MOVE WITH AGV MOVE WITH AGV
1 2	Pallet Box	Storage Stop2	FIRST 1 FIRST 1	MOVE WITH AGV MOVE WITH AGV
1	NutBox	Mach_1	FIRST 1	
1	NutBox	Exit	FIRST 1	
1	Pallet	Storage	FIRST 1	MOVE WITH AGV
1	BoltBox	Mach_2	FIRST 1	
1	BoltBox	Exit	FIRST 1	

The logic loads three boxes of nuts and bolts onto a pallet. It uses the GROUPQTY() function to check if there are any boxes of nuts on the pallet. If there are, it routes the box to the location, Stop1. Once it arrives to the location Stop1, the boxes of nuts (NutBox) are unloaded. This is done with the UNLOAD statement. It only unloads the entities that are boxes of nuts (Unload 3 IFF Entity()=NutBox). The boxes of bolts (BoltBox) remain in the box. We then check to see if the box contains any boxes of bolts. If it does not, the pallet is routed immediately to the original destination, Storage. If it contains boxes of bolts, it is routed to the location Stop2. This example is called Resource.mod and is found in the PROMOD4\MODELS\REFS directory.

7.5 Processing

Processing defines the routing of entities through the system and the operations that take place at each location they enter. Once entities have entered the system, as defined in the Arrivals table, processing specifies everything that happens to them until they exit the system.

Processing is defined in the Processing Editor, which is accessed through the Build menu. This section first describes how to create simple processes, then explains each feature of the Processing Editor.

How To **Create and edit process routings:**

- Select **Processing** from the **Build** menu.

or...

- Right click on the existing process routing and select **Edit**.

7.5.1 Using the Processing Editor

This discussion covers the procedures used to define operations and routings using the Processing Editor. As with most other procedures in ProModel, they may be performed graphically using the mouse, or manually by typing the information directly in the edit tables.

Before you begin to specify the processing logic, define all locations and entities to be referenced in the processing. This is done through the Location and Entity Editors. If you reference a location or an entity that has not yet been defined in a location or entity field, you will be prompted to add that location or entity to the respective location or entity list. However, no graphic gets automatically assigned to the location or entity.

The easiest way to define the processing logic is to define the routing or flow sequence using the tools in the Processing Tools window, which appears in the lower left corner of the Processing Editor. These tools have been designed to allow you to easily and rapidly define the flow of entities through the system. It is also a good idea to define the routing rule for each routing as it is created. Once you have defined the from-to relationships between locations for each entity, fill in the details of the operation and move logic for each location. This is typically done by typing the logic in the operation or move logic column manually or by using the Logic Builder, documented at the end of this section.

Defining processes graphically in ProModel requires interaction with all four process editing windows.

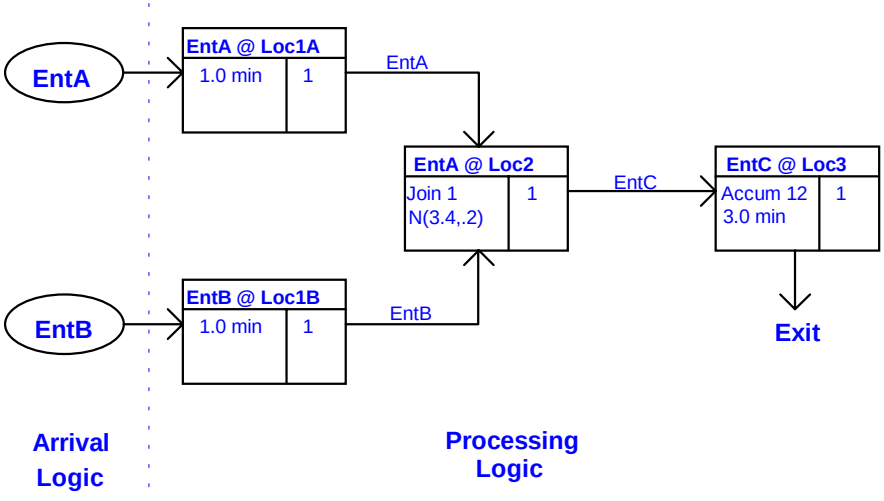
- Process Edit Table
- Routing Edit Table
- Tools Window
- Layout Window

Before discussing the procedures for using these windows interactively, let us look briefly at a process flowchart of a simple model.

Example Model

Two entity types, EntA and EntB, arrive at Locations 1A and 1B, respectively, according to some specified arrival logic. After a short preparation time, both entities are routed to Location 2 where 1 EntB is joined to 1 EntA. At this point the resulting entity, EntC, is sent to Location 3 for consolidation. Twelve EntC's are accumulated at Location 3 and processed together for 3.0 minutes. Then they exit the system.

Process Flow Chart



In the flowchart above, each block represents a process record for an entity at a location. The lower left portion of a block specifies the operation(s) performed on the entity at the location. The lower right portion of the block represents the number of entities output to the next location.

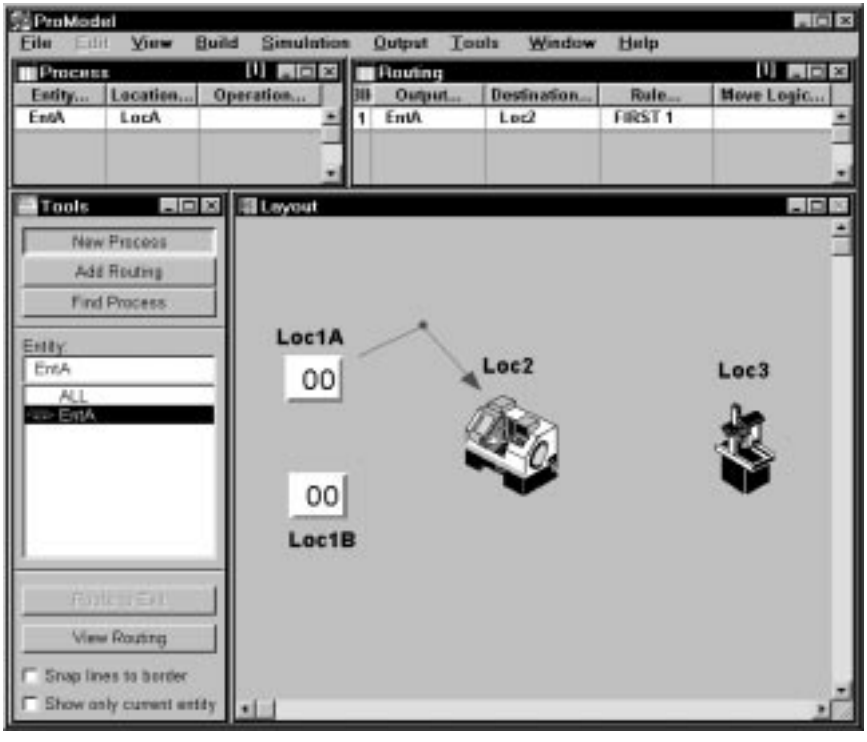
7.5.2 Defining Entity Processing

How To

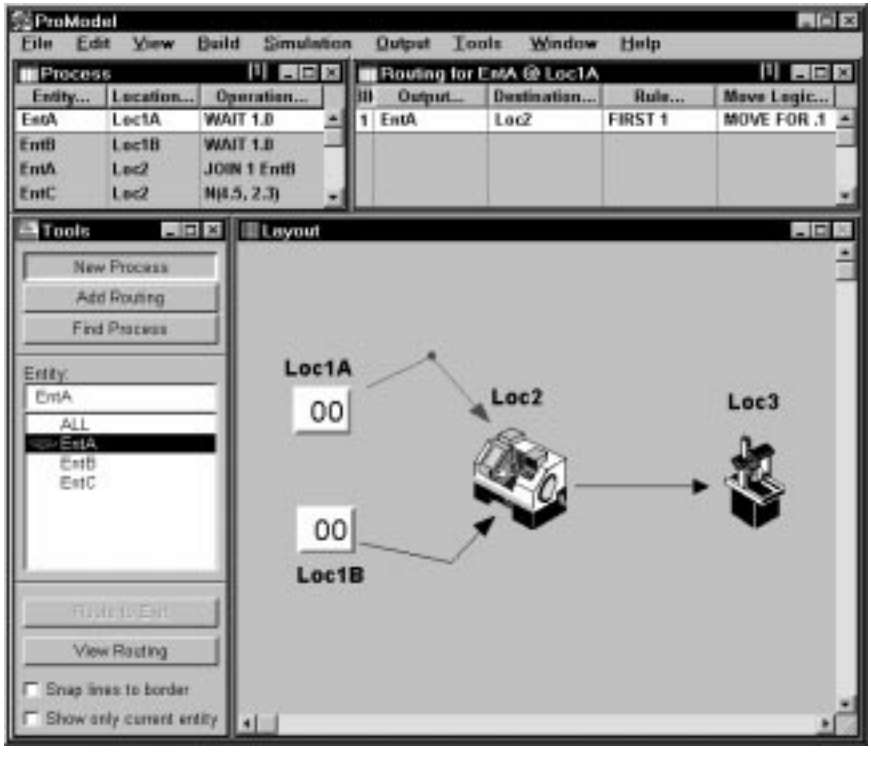
Define entity processing graphically:

1. Select an entity from the entity list in the Tools window. The selected entity will appear in the edit field at the top of the list—this entity will come into the location, it is not the entity that results from the process.
2. Select the desired editing mode: New Process or Add Routing.
3. Click on the first location where the entity will process. A rubber-banding routing line appears. If you select Add Routing, the rubber-banding routing arrow automatically appears from the current location.
4. To choose a different entity as the output entity, select the desired output entity in the tools window.
5. Click the destination location.

In the example below, we first clicked on Loc1A and then on Loc2. The records in the Process and Routing edit tables were entered automatically.



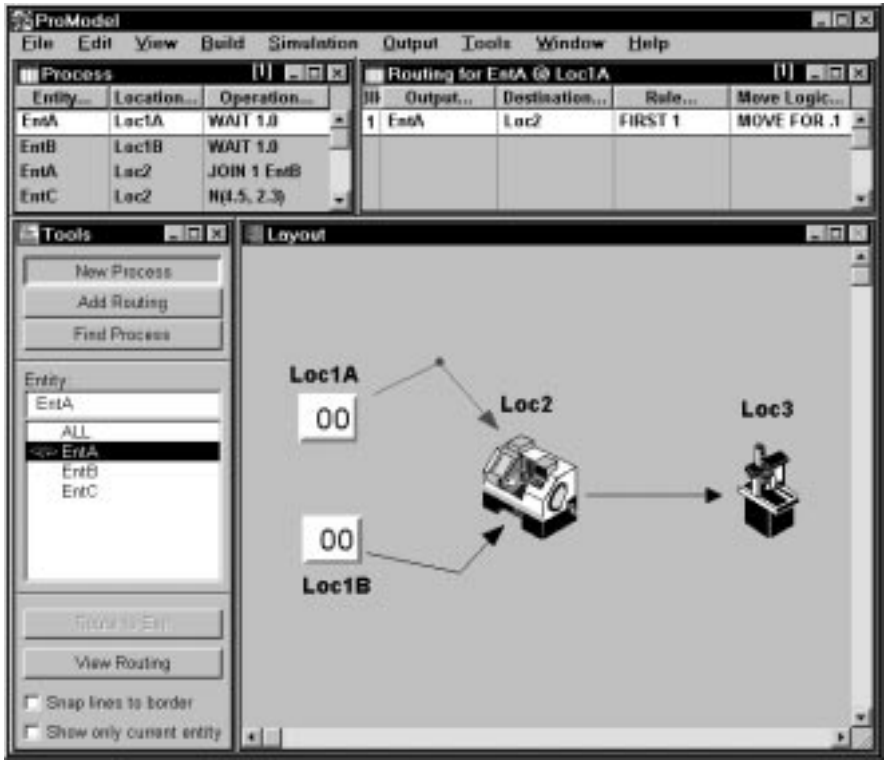
6. Repeat this process until the flow of entities has been completely defined except for exiting the system.
7. From the final processing location route an entity to Exit by clicking on the “Route to Exit” button in the Tools window.
8. Once all routings have been defined, enter the processing logic in the operation field of the Process edit table.



The figure above shows the completed routings for the example model. Note that the operations (For example, Join 1 EntB) have been entered manually in the operation column. These operations could also have been entered by way of the Statement/ Expression Builder, documented at the end of this section.

7.5.3 Processing Editor

The Processing Editor consists of four windows that appear simultaneously, as shown in the following diagram. Although the windows are shown in their default arrangement, you may arrange them as desired.



Process Edit Table Appears in the upper left corner of the workspace and defines the operations performed for all entities at all locations.

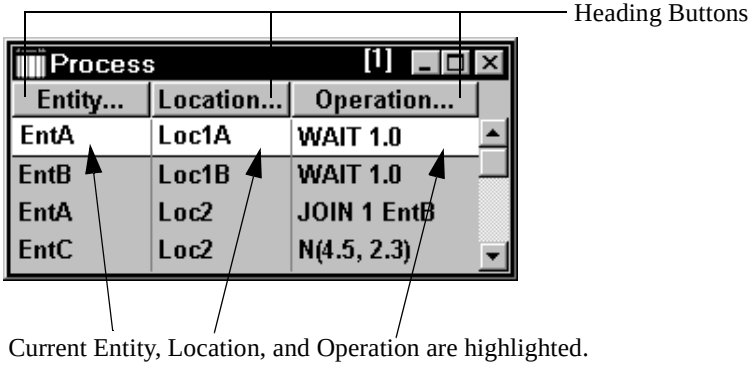
Routing Edit Table Appears in the upper right corner of the workspace and controls the destination of entities that have finished at the location.

Tools Window Appears in the lower left corner of the workspace and is used for graphically defining operations and routings.

Layout Window Appears in the lower right corner of the workspace and shows each location with all from-to routings.

7.5.4 Process Edit Table

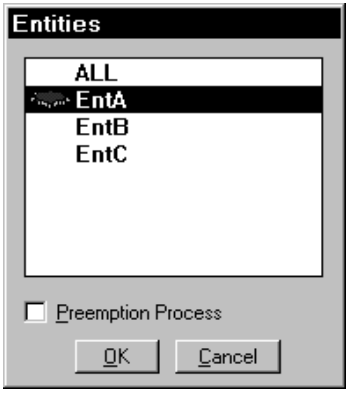
The Process edit table is used to create operation logic for each entity type at each location in the system. Processes for entities at locations may be in any order in the edit table, but for the sake of organization you should group them by entity type or location. The only time the order of processes is significant is when the same entity is routed multiple times through the same location, in which case, later processes must appear somewhere after earlier processes. When searching for the next process, ProModel always searches *forward* in the process list first, and then starts from the beginning of the list.



An explanation of each field of the Process edit table is contained on the following pages.

Entity The entity type for which the process is defined. If all entities at the same location undergo the same operation or have the same routing, the reserved word ALL may be entered in this field. (See discussion on ALL later in this section.) If an entity not previously defined is entered in this field, ProModel will ask if it should create the new entity type.

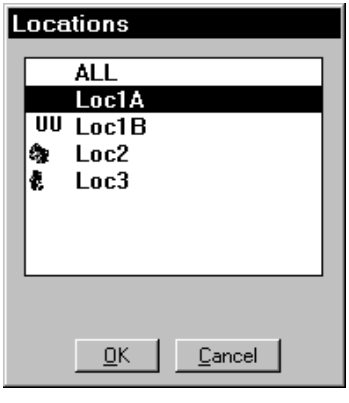
Click on the heading button to bring up the Entities selection box. If this is a preemption process record, check the box (see *Preemption Process Logic* on page 501). Then double-click on an entity to automatically place it in the table. You can also click on an entity name and select OK to place it in the table.



The entity list box defaults to the current field entity, the last entity selected, or the first entity defined.

Location The location where the process occurs.

Click on the heading button to bring up the Locations selection box from which you may choose a location.



The location list box defaults to the current location, the last location selected, or the first location defined.

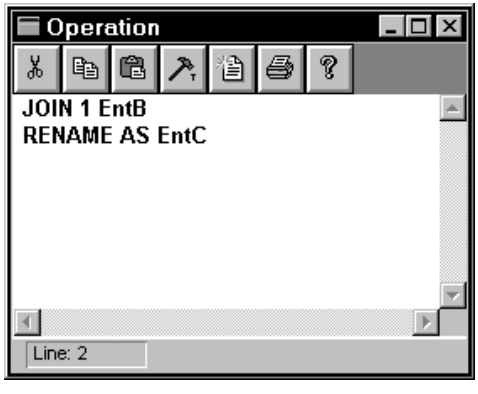
Specifying ALL in the Location field and omitting any routing defines a process for an entity at all locations previously specified as routing destinations for the entity. Because there is no routing, after the entity finishes that process, ProModel will search ahead in the Process edit table for a process for the entity specific to the actual location. The keyword ALL in the Location field is particularly useful when entities route to different locations having the same operations and then route to a common

destination. In most other instances, it is recommended that a subroutine or macro be used to define identical operations.

Operation Operation logic is optional, but typically contains at least a WAIT statement for the amount of time the entity should spend at the location. If the entity needs a resource to process or to be combined in some way with other entities, that would be specified here as well. In fact, anything that needs to happen to the entity at the location should be specified here, except for any information specified in the entity's routing.

Statements can be typed directly into the operation field, or inside a larger logic window after double clicking in the field or clicking on the Operation button. Alternatively, the Logic Builder can help build logic and is accessed by clicking the right mouse button inside the operation field or logic window. All of the statements, functions, and distributions available in the operation field are discussed in detail, including examples, in the *ProModel Reference Guide*.

Each entity performs the operation steps defined for it at a particular location, independent of other operations performed on other entities at the same location.



For more information on using operation logic, see *Operation Logic* on page 499.

Using the “ALL” Entity Type

The reserved word ALL may be entered as the processing entity if all entity types at that location have the same operation. ALL may also be used in the output field of the routing if all entity types at that location have the same routing. If a process record for a location using ALL as the entity follows several process records for the same location using specific entity names, and each process record has a defined routing, the ALL process is interpreted to mean ALL of the rest of the entities. The following examples show how ALL may be used in different situations.

1. All entities have a common operation and a common routing.

To define a common operation and routing for all entity types at a location, simply enter ALL for both the process entity name and the output entity name.

In the following example three entity types, EntA, EntB, and EntC, are all sent to a test station for testing. The test time is .5 minutes and then the entities move on to a packaging station. Though this is a simple example, it shows how one process and routing record is used for all entity types. In contrast, the previous operations at Loc1, Loc2, and Loc3 require separate process and routing records for EntA, EntB, and EntC.

Process Table

Entity	Location	Operation (min)
EntA	Loc1	WAIT N(5,.3)
EntB	Loc2	WAIT U(3,.2)
EntC	Loc3	WAIT T(3,5,9)
ALL	Test	WAIT .5

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Test	FIRST 1	MOVE FOR 1
1	EntB	Test	FIRST 1	MOVE FOR 1
1	EntC	Test	FIRST 1	MOVE FOR 1
1	**ALL**	Packaging	FIRST 1	MOVE FOR 1

2. All entities have common operations but individual routings.

To define common operations but individual routings, use ALL as the process entity and define the common process, but do not define any routings. Then define individual processes for each entity type at the common location, with a blank operation field and the desired routing.

Process Table

Entity	Location	Operation (min)
ALL	Test	WAIT .5
EntA	Test	
EntB	Test	
EntC	Test	

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Pack1	FIRST 1	MOVE FOR 1
1	EntB	Pack2	FIRST 1	MOVE FOR 1
1	EntC	Pack3	FIRST 1	MOVE FOR 1

If only the destination is different, but move times and output quantities are identical, an alternative method is to assign each entity an attribute that corresponds to the destination's name-index number and then route with the LOC() function as shown in the following example.

Process Table

Entity	Location	Operation (min)
ALL	Test	WAIT .5

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	**ALL**	Loc(Att1)	FIRST 1	MOVE FOR 1

3. All entities have a common routing but individual operations.

To define individual operations along with a common routing for all entity types at a location, define operations for each entity, but do not define any routings. Then define a process record for ALL at this location and define the common routing for all entity types.

Process Table **Routing Table**

Entity	Location	Operation (min)	Blk	Output	Destination	Rule	Move Logic
EntA	Loc1	WAIT .4					
EntB	Loc1	WAIT .5					
EntC	Loc1	WAIT .6					
ALL	Loc1		1	**ALL**	Packaging	FIRST 1	MOVE FOR 1

Alternatively, you can assign an attribute to each entity, representing the processing time or some other entity-specific parameter. Then use the attribute as the processing time, or call a subroutine and pass the attribute as a parameter for entity-specific processing.

In the following example, the test time for each entity type is different. This time is stored in an attribute, Oper_Time. The attribute is then listed on a line in the operation logic with a WAIT statement to signify an operation time. Once the test time for each entity is completed, the entities are all routed to a packaging location.

Process Table **Routing Table**

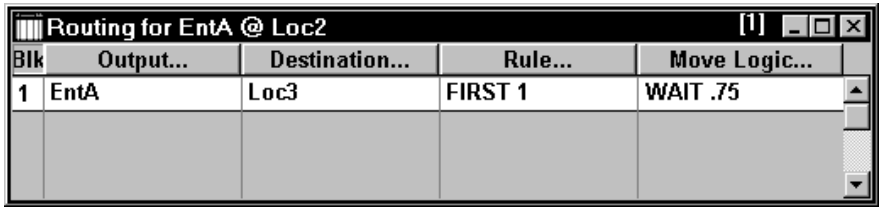
Entity	Location	Operation (min)	Blk	Output	Destination	Rule	Move Logic
ALL	Test	WAIT Oper_Time	1	**ALL**	Packaging	FIRST 1	MOVE FOR 1

7.5.5 Routing Edit Table

The Routing edit table defines the outputs for each process record defined in the Process edit table. The Routing edit table is really just a sub-table to the Process edit table (i.e., all routings that appear in the routing edit table apply to the currently highlighted process), though the two tables appear side by side. *Not all process records need to have a corresponding routing*. If the routing is omitted, ProModel will search forward in the Process edit table for another process for that entity at that location. So an entity's complete processing at a location could be broken into several records. In that case, only the last process would have a routing. If no routing is defined for at least one of the process records for a given entity and location, an error occurs.

Another situation that does not require routing is when an entity changes its name at a location after a RENAME AS or SPLIT AS statement. Any time during processing logic that an entity changes its name, ProModel searches forward in the Processing

edit table until it finds a process for the new name at the same location. For example, if the identity of an entity is changed through a RENAME AS statement in the operation logic, then no routing block will apply to the old entity. Instead, the newly named entity will be routed by the process for the new name. (See *Rename* on page 209 of the *ProModel Reference Guide*).



The fields of the Routing edit table are as follows:

Blk This field contains the block number for the current routing block. A routing block consists of one or more alternative routings from which one is selected based on the block rule (e.g., a list of routings where one is selected based on the most available capacity). Since all of the routings using the same rule are part of the same block, only the first line of each routing block contains a route block number. If no routing blocks have been referenced explicitly in the operation logic (for example "ROUTE n"), *all* routing blocks will be executed in sequence upon completion of the operation logic. See *Route on page 221* of the *ProModel Reference Guide*. Multiple routing blocks are processed sequentially with the next block being processed when all of the entities in the previous block have begun executing any move logic defined. To change the routing block number or add a new routing block, see the discussion on the Routing Rule dialog box later in this section.

The following example shows a process record with two separate routing blocks. Note that *both* routings will execute upon completion of the operation time because no ROUTE statement has been specified. One EntB gets routed to Loc2 and one EntC gets routed to Loc3.

Process Table

Entity	Location	Operation (min)
EntA	Loc1	WAIT N(5,.3)

Routing Table

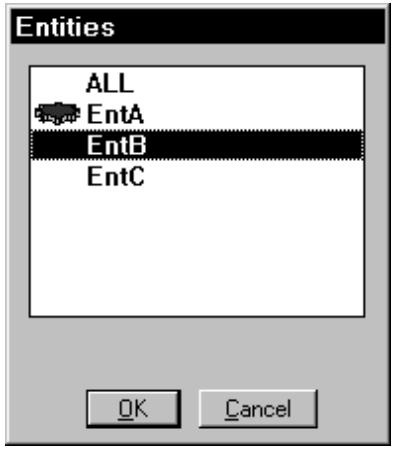
Blk	Output	Destination	Rule	Move Logic
1	EntB	Loc2	FIRST 1	MOVE FOR 1
2	EntC	Loc3	FIRST 1	MOVE FOR 2

Output If a routing is defined, the name of the entity resulting from the operation must be entered here. This name may be the same as the entity that entered, another name, or even several names, each on a different line.

Using another name works much like a RENAME AS statement, except that the entity is routed according to the routing block instead of being processed further at the same

location. Using several names on different lines is similar to having a SPLIT AS statement in conjunction with a RENAME AS statement. The difference is that multiple routing blocks are processed sequentially while split entities get processed concurrently.

If the reserved word ALL was used as the incoming entity type for this process, it may also be entered here. Otherwise, every entity entering the location will change to the specified output entity. (See the discussion on using ALL in the Entities section.)



The entity list box defaults to the current field entity, the last entity selected, or the first entity defined.

To better anticipate the entity entry that is likely to be made, the entity highlighted in the list box defaults, in order, first to the current field entity, if any, then to the last entity selected, if any, and finally to the first entity defined.

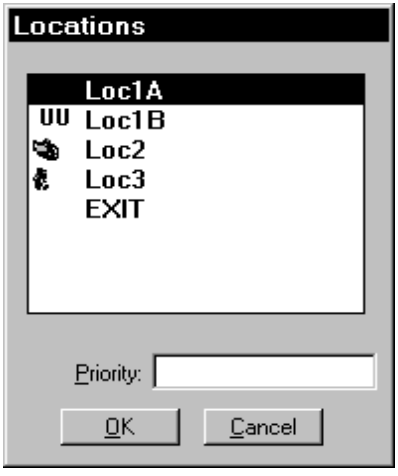
The example below shows how an incoming entity, EntA, changes identity and becomes an EntB upon exiting location Loc1. This is done by simply specifying the new entity name as the output entity.

Process Table **Routing Table**

Entity	Location	Operation (min)	Blk	Output	Destination	Rule	Move Logic
EntA	Loc1	WAIT T(2,5,8)	1	EntB	Loc2	FIRST	MOVE FOR 1

Destination This is the location to which entities move after the operation is complete. Optionally, the destination may be followed by a comma and a routing priority. If no priority is specified, the default is zero.

The location name may be typed directly in the field, or selected by either clicking in the field and then on the heading button, or double clicking in the field. The resulting dialog box looks like this:



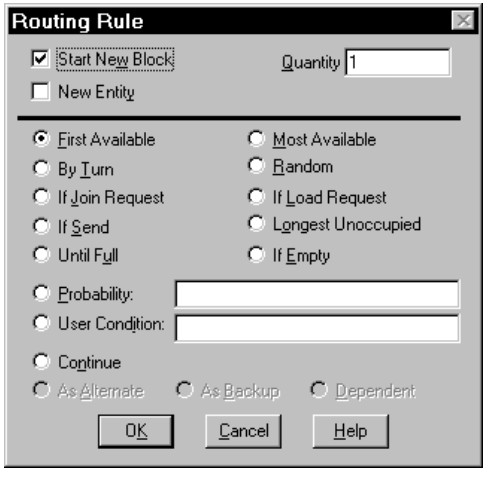
Instead of entering a location name, you may use the LOC() function which returns the location name corresponding to the index designated by an expression. For example, LOC(5) routes to the fifth location defined or the fifth record in the location edit table. You can also specify variable destinations such as LOC(Att4). However, the LOC() function may not be used in conjunction with the BY TURN, UNTIL FULL, CONTINUE, and IF EMPTY routings. (See *Loc()* on page 180 of the *ProModel Reference Guide* for more information.)

Rule This field defines the rule for selecting the route destination. An output quantity may also be specified as part of the rule. This quantity works much life the SPLIT AS statement. (See the following discussion on the Routing Rule dialog box.) To open the Routing Rule Dialog box, double click in the field or click in the field and then on the heading button.

Move Logic The Move Logic window allows you to define the method of movement as well as any other logic to be executed between the time the next location is claimed for routing to the time the entity arrives, but not yet enters, the next location. To open the Move Logic window, double click in the field or click the heading button. This window allows you to manually edit the logic or click on the Build button to use the Logic Builder. It also provides other convenient buttons for editing and printing the move logic. For single move statements it is easiest to right click in the move field to open the Logic Builder. If left blank, it is assumed that no time, resources, or path networks are needed to make the move. (See *Routing Move Logic on page 310*.)

Routing Rule Dialog Box

The Routing Rule dialog box provides methods for selecting an entity's destination after finishing a process. The Rule heading button in the Routing edit table, brings up the Routing Rule dialog box. The fields of this dialog box are defined in the following example.



Start new block Check this box to signal the beginning of a new routing block. Checking this box will place a number in the Blk field for that record.

New Entity Check Box In order to let you designate whether a routing block applies to the main entity or if you wish to create a new entity, ProModel includes a New Entity check box in the routing rule dialog. By default, ProModel does not check the first routing block for a process record but does check subsequent blocks unless they share the same input and output name. If you check the New Entity box, an asterisk (*) appears after the block number in the block column of the routing table.

If you *check* the new entity box, the entity using the routing block will begin routing with new cost and time statistics. If the new entity box remains *unchecked*, ProModel assumes the entity routing from this block is a main entity (the parent entity), carrying with it all cost and time statistics.

If the new entity field remains unchecked and you enter a quantity field value greater than 1, ProModel behaves just as it does with a SPLIT AS statement, dividing the cost statistics between the split members and resetting all time statistics to zero.

Note A run-time error occurs if you fail to route one (and only one) main entity for a process. If no routing block executes for the main entity, a “No routing defined for main entity” error occurs. If more than one route block executes for the main entity, a “Main entity already routed” error occurs.

Quantity The number of entities resulting from this routing block. The default is one. Several entities of the same name can be created from a single entity, much like a SPLIT AS statement, by entering a number greater than one. Each of the new entities then processes the routing block one entity at a time. For example, suppose an entity called bar_stock enters a location and is cut into 10 smaller segments, called bars, for production. Enter 10 as the quantity and bar as the output entity, as in the example below. Only the first line in a routing block can specify a quantity.

Process Table **Routing Table**

Entity	Location	Operation (min)	Blk	Output	Destination	Rule	Move Logic
bar_stock	Loc1	WAIT T(2,5,8)	1	bar	Loc2	FIRST 10	MOVE FOR 1

Routing Rules Choose the rule for selecting the next location. Only one rule may be chosen for each routing line. Note that the last three routing rules, alternate, backup and dependent, may not be chosen as the start of a new routing block. Note also that no more than one of the other rules can appear in a single block (e.g., you cannot mix a First Available rule and a Most Available rule in the same block).

For exact syntax and examples of each routing rule, see *Routing Rules* on page 91 of the *ProModel Reference Guide*.

Routing Move Logic

The Move Logic window allows you to define the method of movement as well as any other logic to be executed prior to or after the move actually takes place.

Once the route condition or rule has been satisfied for allowing an entity to route to a particular location, the move logic is immediately executed. The entity does not actually leave the current location until a move related statement (MOVE FOR, MOVE ON, MOVE WITH) is executed or the move logic is completed, whichever happens first. This allows the entity to get one or more resources, wait additional time, or wait until a condition is satisfied before actually leaving the location.

Any statements encountered in the move logic after the move related statement are executed after the move is complete but before the entity actually enters the next location. This is often useful for freeing multiple resources that may have been used to transport the entity.

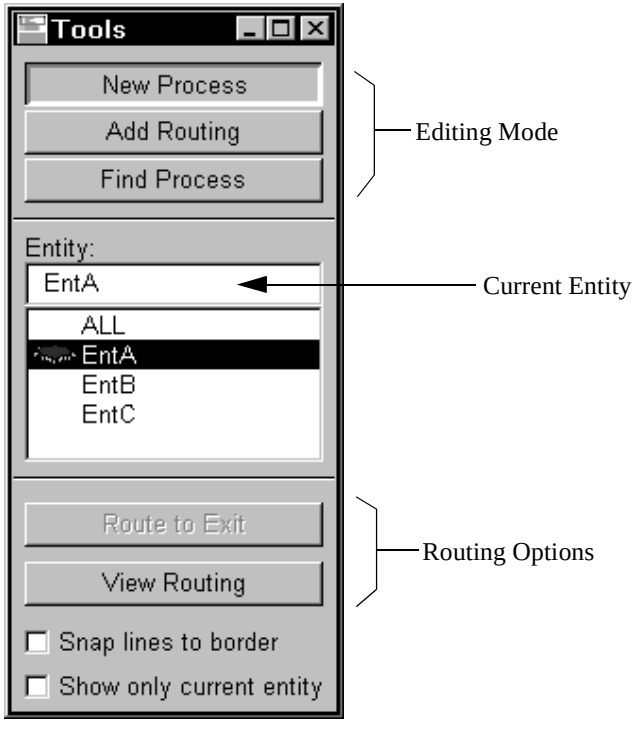
When defining exit logic, such as incrementing a variable used to track the number of exits from a location, it can generally go before the move statement unless a MOVE WITH statement is used and the entity must capture the resource before making the move. In this situation a GET statement should be specified first to get the resource. Then the exit logic may be specified followed by the MOVE WITH statement.

Any delay occurring as a result of move logic is reported as part of the entity's move time. For more information see *Routing Move Logic* on page 505.

7.5.6 Processing Tools

The Tools window provides graphical aids that may be used to define processing records and routing records. It is also used to define the graphical paths that entities follow when moving without a path network between locations.

The Tools window, which appears along with the other Processing windows, can define processing in one of two modes, New Process or Add Routing. Each is explained next. Additionally, Find Process Mode is available. To select a mode, click on the desired button. Each of the modes is described on the next page.



In addition to option buttons (New Process, Add Routing, or Find Process) for the process editing mode, the Tools window contains a list of defined entities as well as the reserved word ALL to represent all entity types. The entity in Processing Tools applies to either the process or routing entity, depending on what is currently being defined.

New Process Mode

New Process Mode is used to create a new process record. A new process is automatically created for the selected entity each time you click on a location.

This mode should be used if you want to create a process for a particular entity at a location. You may even create multiple processes for the same entity and location if you want to re-route an entity through the same location more than once for additional operations. Once a new process is created, the mode automatically changes to Add Routing mode to enable a routing to be defined for the process.

How To Define a new process using the Tools window:

1. Depress the New Process button.
2. Select the entity for which a new process is to be defined from the Tools window.
3. Click on the location where the entity will be processed in the layout window. A new process record is created in the edit table. The mode is automatically switched to Add Routing mode and a rubber banding line appears that connects the mouse pointer to the location.
4. If a different entity is to be output from the process, select it from the Tools window.
5. Click on the destination location. A new routing record appears in the edit table and the mode switches back to New Process mode.

How To Delete a process or routing record:

1. Click inside the desired record in the Process or Routing edit table.
2. Select **Delete** from the **Edit** menu.

Editing a Routing Path

Once a routing path has been defined you may edit the path (regardless of the current mode) by clicking anywhere on the routing path. This selects the path, and allows you to change the source or destination of the routing by dragging the beginning or end of the path to a new location. It also allows you to move any intermediate joint in the path to change the shape. You may also click on a path with the right mouse button to create or delete a joint.

If a process is already defined and a location is moved while in the Location module, the connecting leg of any routing lines will also move.

Add Routing Mode

Add Routing Mode is used to create multiple routings for a single process record. Suppose an entity, EntA, can travel to one of three locations depending on which is available first. Selecting the New Process mode and then defining the entity process causes the entity to travel from one location to another location. Selecting Add Routing mode afterwards allows you to define a different destination location within the same routing block.

How To **Add additional routings to an existing routing block:**

1. Select the process record in the Process edit table that needs an additional routing line (you may use the Find Process button to locate the process record).
2. If you wish to insert the routing record rather than simply append the record to the current routing list, highlight the routing record where the routing is to be inserted and choose **Insert** from the **Edit** menu.
3. Select the Add Routing button from the Tools window. A rubber-banding line is created.
4. Select the entity in the Tools window to be output in this routing.
5. Click on the desired destination location. This creates a new routing record in the Routing edit table

Note To cancel a routing once a rubber-banding line appears, click on the New Process button or click on the originating location.

How To **Route a current entity to exit:**

1. Click the Add Routing button.
2. From the Tools window, select the entity for which you want to define a new process.
3. Click the Locations button in the process record dialog and select the location from which the entity will exit the system. ProModel creates a new process record in the edit table and displays a rubber banding line from the selected location.
4. Add joints as needed to define the exit path.
5. Click the Route to Exit button in the tools dialog.

Find Process Mode

To find a previously created process for an entity type at a certain location, use Find Process mode.

How To **Find a process for an entity at a location:**

1. Click on the Find Process button.
2. Click on the desired entity type.
3. Click on the desired location. The first process found for that entity type at the location will be highlighted in the Edit window.

This option always searches forward in the process list. By clearing the entity field in the tools window, the next process for any entity at the location selected is found. If multiple process records exist for the same entity and location, the edit table indexes forward to the next process record each time you click on the location.

View Routing

The View Routing button in the Processing Tools window causes the process record currently highlighted in the edit table to become centered on the layout.

Snap Lines to Border

When you check this option, ProModel snaps the routing lines to the location's bounding area rather than a specific position on the graphic.

Show Only Current Entity

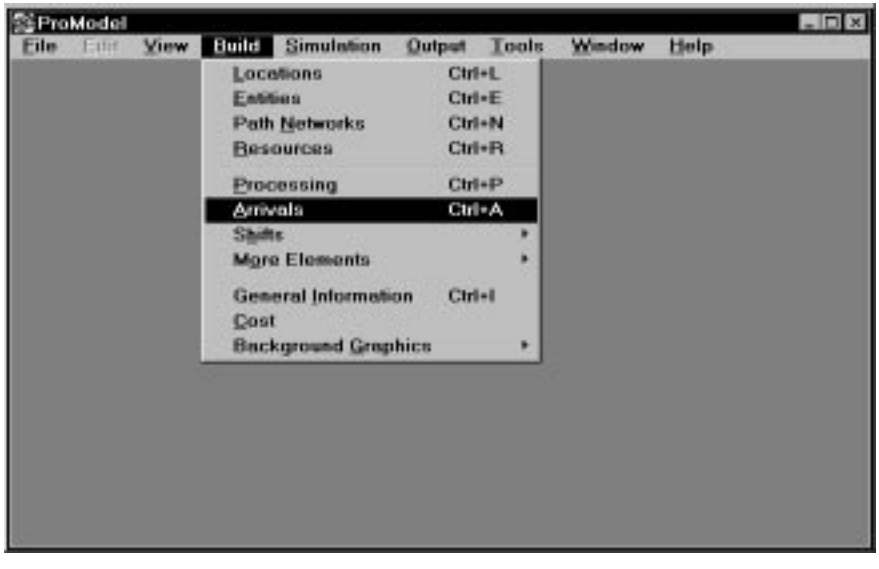
When you highlight an entity in the list box, checking this option will show only those routings associated with the entity.

7.6 Arrivals

Any time new entities are introduced into the system, it is called an arrival. An arrival record is defined by specifying the following information:

- Number of new entities per arrival
- Frequency of the arrivals
- Location of the arrival
- Time of the first arrival
- Total occurrences of the arrival

Any quantity of any entity type can be defined as an arrival for a location. The frequency of arrivals can be defined as either a distribution or as an arrival pattern which cyclically repeats over time.

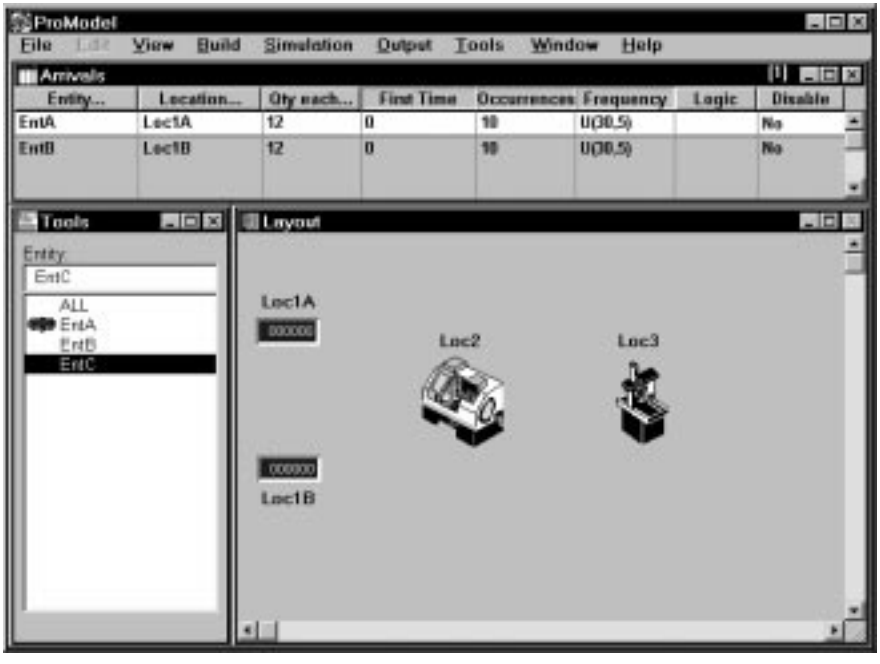


How To Edit Arrivals

- Select **Arrivals** from the **Build** menu.

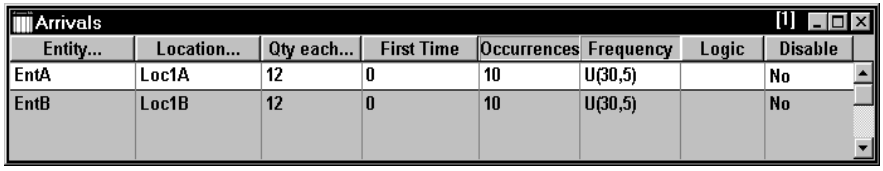
7.6.1 Arrivals Editor

The Arrivals Editor consists of three windows that appear on the screen together. The Arrivals edit table contains the specifications of each arrival to the system and appears across the top of the screen. The Tools window contains tools for defining arrivals graphically and appears at the bottom left corner of the screen. The Layout window appears in the lower right corner of the workspace.



7.6.2 Arrivals Edit Table

The Arrivals edit table lists all scheduled entity arrivals to the system. The various fields are explained below.



Entity...	Location...	Qty each...	First Time	Occurrences	Frequency	Logic	Disable
EntA	Loc1A	12	0	10	U(30,5)		No
EntB	Loc1B	12	0	10	U(30,5)		No

Entity The name of the arriving entity.

Location The name of the location where the entity is to arrive.

Qty each... The number (1 to 999999) of entities to arrive at each arrival time interval. Any valid expression may be entered here except for attributes and non-general system functions. This field is evaluated throughout the simulation run and will change if the result of the expression changes.

To fill a location to capacity at every arrival time, use the keyword INFINITE, abbreviated INF.

If you have previously created an arrival cycle and want to use it for this arrival, enter the name of the arrival cycle followed optionally by a quantity. You may also click on the Qty each... heading button to select from the list of defined cycles. See the section on Arrival Cycles for more information about defining cycles.

First Time This option allows you to dynamically vary the time of the first arrival to your model. You may define scheduled arrivals to occur at given intervals (e.g., appointments) or use an arrival cycle to define random arrivals over a period of time (this value is the start time for the first cycle). ProModel evaluates this field only at the beginning of the simulation.

Occurrences The number of times per simulation run that ProModel will generate arrivals (1 - 999999). Entering the reserved word INFINITE (abbreviated INF) will cause ProModel to send the specified number of arrivals at every arrival time without limit. This value may be any expression and is evaluated only at the beginning of the simulation. If an arrival cycle is used, this is the number of times to repeat the cycle.

Frequency The inter-arrival time or time between arrivals. Any valid expression may be entered here except for attributes and non-general system functions. If an arrival cycle was entered for the arrival quantity, this is the time between the start of each cycle. This field is evaluated throughout the simulation run and will change if the result of the expression changes.

Logic This field defines any optional arrival logic, consisting of one or more general statements, to be executed by each entity upon its arrival (e.g., assigning attribute values to entities as they arrive). Double click inside this field or click the logic button at the top of the column to define logic for an arrival.

Disable Set this field to YES or NO if you want to temporarily disable this arrival without deleting it. This is useful when debugging a model and for verification purposes where you want to follow a single entity through the system.

Note

Arrivals edit table notes:

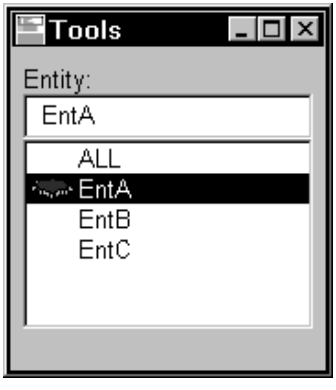
1. When several different entity types are scheduled to arrive at a location simultaneously, they will arrive in the order they are listed in the Arrivals table. To have them alternate their arrivals, enter a 1 in the "Qty each" field and the total entry quantity in the "Occurrences" field.
2. Arrivals defined through an external arrival file will be appended to the arrival list. Therefore, if an external arrival file is the only source of arrivals, the Arrival edit table may be left blank. See the section on External Files for more information on arrival files.
3. If the capacity of the location is insufficient to hold all the arriving entities, the excess entities are destroyed. Therefore, the arrival location should have a capacity at least equal to the "Qty each" in the Arrivals edit table. If more entities are scheduled into the system than are exiting, the arrival location may not have enough capacity to handle all the arrivals.

7.6.3 Defining Entity Arrivals

Arrivals may be defined graphically by using the tools in the Tools window, or by manually entering the arrival information directly in the Arrivals edit table.

How To Define arrivals graphically:

1. Select **Arrivals** from the **Build** menu.
2. Select the desired entity from the Tools window.



3. Click in the layout window at the location where the entity is to arrive. (You may need to scroll through the layout to bring the desired location into view.)
4. Enter the specifications for the arrival record (e.g., arrival quantity, frequency, etc.).

How To Define arrivals manually:

1. Select **Arrivals** from the **Build** menu.
2. Enter the Entity, Location, and Quantity through either the keyboard or by clicking on the respective heading buttons and choosing the proper information.
3. Enter the First Time, Occurrences, and Frequency through either the keyboard or by clicking the right mouse button inside the desired field and using the statement builder.
4. Click on the **Logic** or **Notes** heading button to enter desired logic or notes.

7.6.4 Independent Arrivals

An independent arrival is any arrival assigned to occur at a specific time or at a fixed interval. Independent arrivals include such things as order entries and pickup and delivery times. When defining independent arrivals, remember that simulation can model only *predefined* arrival schedules. This means that dynamically scheduled arrivals must take place where you define the arrival schedule.

When defining independent arrivals, you may:

- Define them by elapsed time, day and time, or calendar date.
- Assign them to occur at fixed intervals (e.g., trucks arriving every three hours).

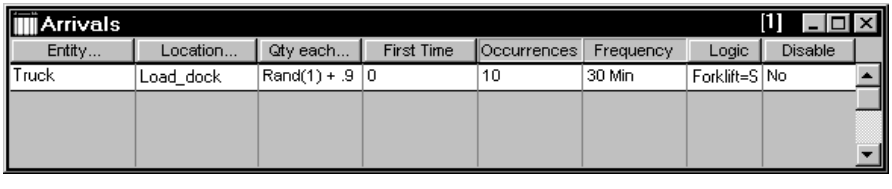
Note When you define independent arrivals as intervals, the next arrival time is independent of the previous arrival. For example, trucks will arrive at the loading dock based on the clock time—*not* the time elapsed since the last arrival.

- Allow a positive or negative offset to adjust the scheduled time of arrival (e.g., trucks must arrive at the loading dock at least ten minutes early).
- Define a distribution to allow variability from the adjusted arrival time.
- Allow the possibility that an entity will not arrive at all.
- Define specific arrivals for certain resources or resource types.

How To

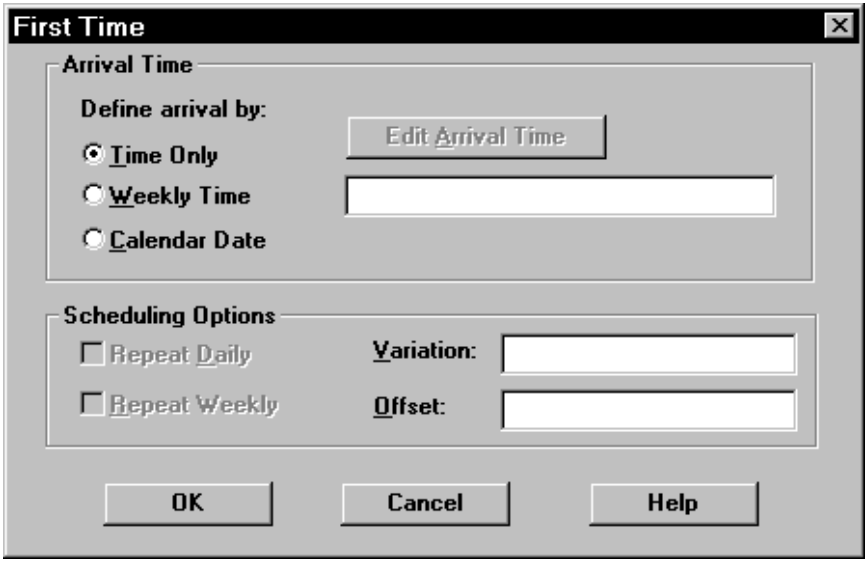
Define independent arrivals:

1. Open the **Arrivals** module from the **Build** menu.
2. Click on the **Entity** button and select the entity type you want to schedule (e.g., truck, client, or shipment).
3. Click on the **Location** button and select the location where you want the independent arrival to appear (e.g., specific bay of a loading dock).
4. Click on the **Qty Each** button and enter the number of entities to arrive at the scheduled time. To model *no shows*, enter the expression "Rand(1) + <*probability of showing*>." For example, if only 90% of scheduled deliveries show up, enter "Rand(1) + .9."

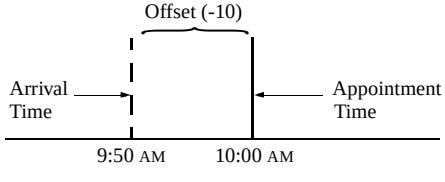


5. Click on the **First Time** button to open the dialog used to define the independent arrival time for the entity. If you define a block of identical deliveries arriving at equal intervals, this is the time of the first delivery in the block. (You may enter the arrival time by elapsed time—since the start of the simulation—by a weekday and time, or by calendar date.)

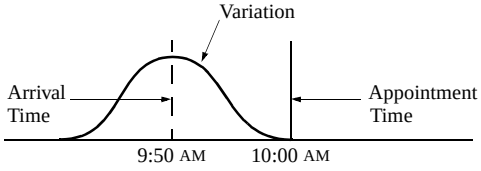
Note The arrival time must match the time units selected for the simulation run-length. If you defined the elapsed time by calendar date, you must also define the simulation length by calendar date. To edit arrival times, select either day and time or calendar date and click the Edit Arrival Time button.



6. In the offset field, enter any offset you wish to apply to the arrival time. This will direct incoming deliveries to arrive earlier (or later) than scheduled. For example, if a delivery at a loading dock is to arrive 10 minutes prior to the scheduled delivery time, enter "-10."



7. In the variation and offset fields of the First time dialog, enter any optional distribution to define the variation from the adjusted arrival time. Generally, to prevent the arrival from being unrealistically late, you should use a doubly-bound distribution (i.e., uniform, normal, triangular, or beta). The triangular and beta distributions provide the most realistic variation.



8. In the Occurrences field of the arrivals edit table, enter the number of times to repeat this delivery definition. (Enter 1 if it will occur only once.)

Arrivals [1]

Entity...	Location...	Qty each...	First Time	Occurrences	Frequency	Logic	Disable
Truck	Load_dock	Rand(1) + .9	0	10	30 Min	Forklift=S	No

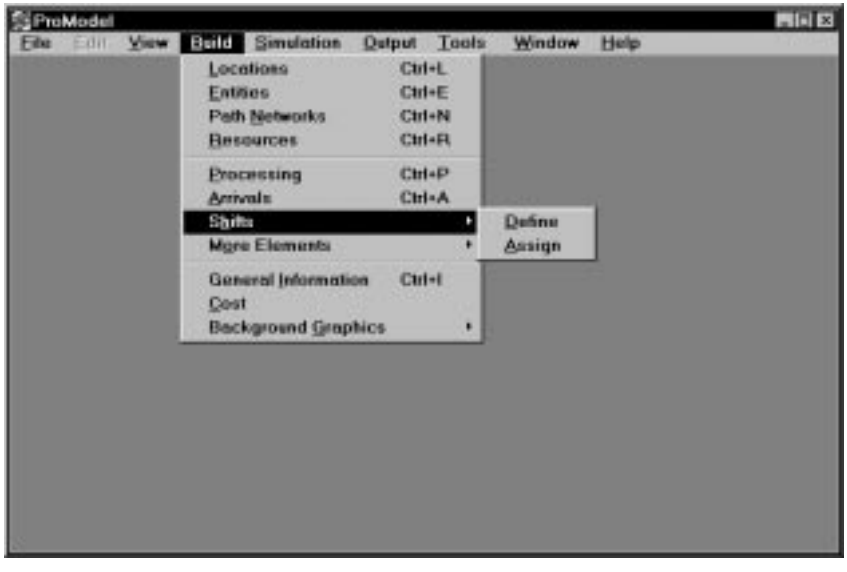
9. If the number of occurrences you entered is greater than zero, enter a time interval in the frequency field. ProModel assumes this interval to be fixed for each occurrence (if you enter an expression, ProModel evaluates it only once and applies it to each occurrence). The distribution defined in the variation field of the First Arrival dialog applies to each occurrence.

10. In the Logic field of the arrivals edit table, enter any attribute assignments you will use to determine the processing of the entity. This might include the resource or entity type (e.g., order number or shipment tracking) and it will be helpful if you define these attributes beforehand.

7.7 Shifts & Breaks

7.7.1 Shift Definition

Weekly shifts and breaks for locations and resources are defined using the Shift editor and may start and end at any minute of the day. Shifts and breaks are defined by selecting blocks on a grid divided into days and hours. Once a weekly shift and break schedule has been defined, it may be saved in a shift file, with an SFT extension.

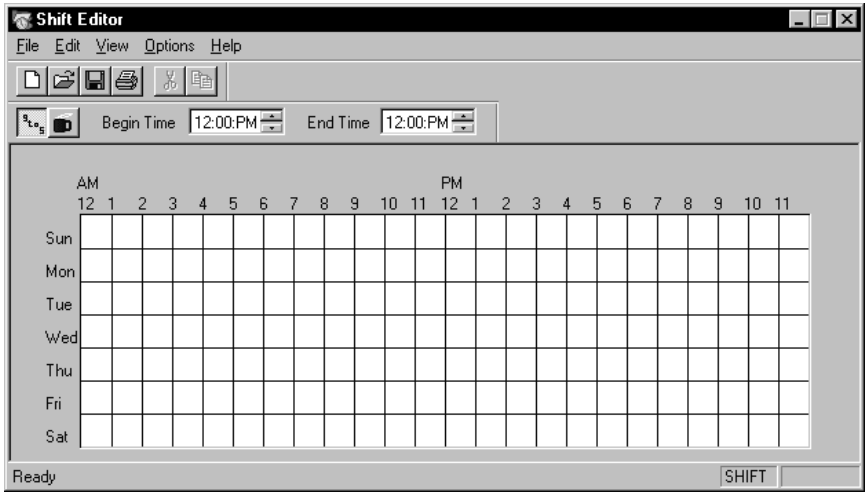


How To **Define a shift:**

1. Select **Shifts** from the **Build** menu.
2. Select **Define** from the submenu.

Shift Editor

The Shift editor window consists of a menu bar, Shift and Break mode buttons, time control buttons, and a grid representing one week of time.



The remainder of this section describes the Shift Editor menus and the following procedures:

- Drawing a Block of Time for a Shift or Break
- Selecting a Block
- Resizing a Block
- Editing the Begin or End Time
- Deleting a Block
- Duplicating a Specific Day's Shift
- Customizing Shift and Break Colors

Shift Editor Menus

The menus used in the Shift editor are accessible from the menu bar at the top of the editor and include the following:

File For opening and saving shift files.

Edit For deleting unwanted shift and break blocks. You may also delete or duplicate a specific day of the shift. If you delete a shift, the shift as well as the breaks in the shift are deleted.

Options For customizing the colors representing shifts and breaks.

Drawing a Shift or Break Block

When drawing a shift or break block, you must be sure to follow these rules:

- Shift blocks may not overlap other shift blocks.
- Break blocks may be drawn only on top of shift blocks.
- Break blocks may not overlap or be adjacent to other break blocks.

How To **Draw a block:**

1. Click on the **Shift** or **Break** button to designate the type of block.
2. Click and begin dragging the mouse from the day and time on the grid the block should begin.
3. Release the mouse button at the time the block should end.

If you want to define a block more precisely than the grid allows, see Editing the Begin or End Time below.

Caution If the block you draw is invalid, it will not appear on the time grid.

Selecting a Block

To edit an existing block, you must first select it.

How To **Select an existing block:**

- Click on the block. (A border appears to show that the block has been selected.)

How To **Deselect a block:**

- Click on the selected block or click on the white area of the window.

Resizing a Block

How To **Resize a block:**

1. Select the block.
2. Drag the border of the block until the block is the desired size.
3. Release the mouse button.

Editing the Begin or End Time

Time blocks for a shift may be created in one minute intervals. It is difficult to graphically define a shift this precise, so there is the option to edit the begin or end time of a shift numerically.

How To **Edit a block's begin or end time:**

1. Select the block.
2. Adjust the begin or end time accordingly using the buttons at the bottom of the screen.
3. Click on the **Update** button.

Deleting a Block

There are three ways to delete a block. Use the one easiest for you.

How To **Delete a block:**

- Select a block and choose the Delete option from the Edit menu.

or...

- Select a block and press the Delete key on the keyboard.

or...

- Select a block and size it down until the highlight box is gone.

Duplicating a Specific Day Shift

In many instances, the shift schedule for a specific day of the week is identical to shift schedules for other days of the week. Rather than creating the same shift block for several days of the week, it is possible to duplicate one day's shift block to another day of the week.

How To Duplicate a shift block from one day to another day:

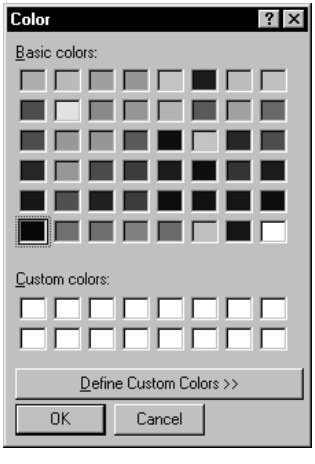
1. Select any shift or break segment for the day you wish to duplicate.
2. Select **Duplicate** from the **Edit** menu. The day of the week you have selected to duplicate is displayed in the Edit Menu of the Shift Editor. For example, if you selected the shift block for Tuesday, the editor will display "Duplicate Tuesday" in the Edit Menu.
3. In the shift schedule, click the mouse on the day you wish to copy the shift block to.

Customizing Shift and Break Colors

The colors that represent shifts and breaks can be customized.

How To Change the color for shifts or breaks:

1. Choose Colors under the Options menu. The colors dialog box will appear.



2. Click on the **Shift** or **Break** button.
3. Click on the desired color.
4. Click **OK**.

7.7.2 Shift Assignments

The Shift Assignment module allows you to model everyday, real-life situations involving scheduling and availability issues, and you can easily define logic to control the way your model handles these problems.

If you have an employee that works a split shift, just assign two shifts to the resource with the corresponding start times. If you have a processing location that can only be used during specifically scheduled hours, set up a separate shift for that location.

If you have a plant or some other operation just starting up, and you need to run a specific shift for the first week and another for two more weeks before going to your full capacity shift schedule, set up three shifts and assign them all to the plant in one step, indicating the appropriate start times for each.

Or if you want to establish a controlled location gateway: Among fulfilling other duties, your employee needs to begin doing something (processing a certain entity at a certain location) at a certain time, so you set up a queue location, a gate location (with a capacity of one), and a processing location. Assign a shift to the gate location so it will come on line at the designated time. Now the gate location begins taking the entities from the queue location and moving them to the processing location, where the employee will be requested at the appropriate priority level.

There are many ways to use shift assignments and shift logic to solve any number of problems in creating a valid model. This chapter explores the features and functionality of the Shift Assignments module, including statements and functions for shift and break logic.

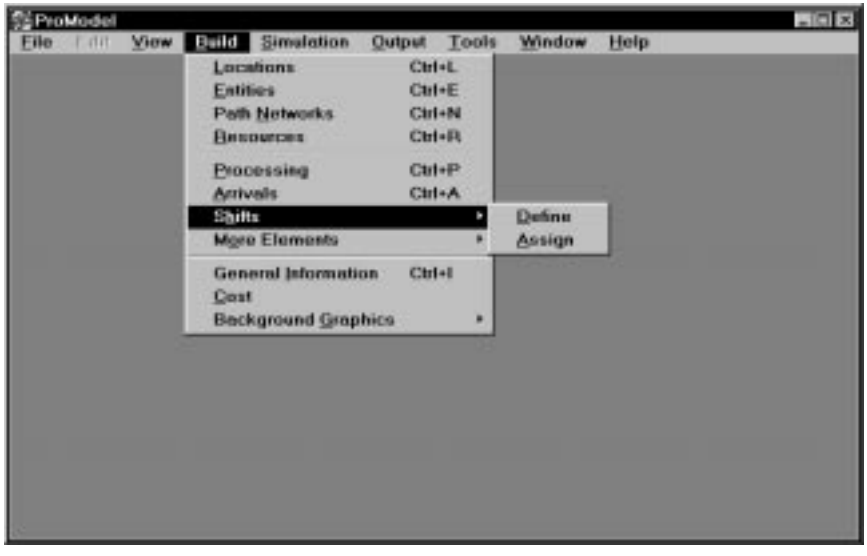
Assigning Shifts

ProModel allows you to select multiple locations and resources and assign them to a single shift record. You can also:

- assign a location or resource to multiple shift files with a start time for each shift,
- define off-shift and break priorities, and
- create off-shift and break logic.

The Shift Assignments module allows you to schedule the availability of resources and locations based on shifts and work breaks defined in the shift editor. When a location or resource goes off shift or on break, it is off-line or off-duty and is reported in the output statistics as non-scheduled time rather than downtime.

The off-shift and break logic are optional and allow you to control more precisely when a resource or location may go off shift, on break, and how long before it becomes available again.

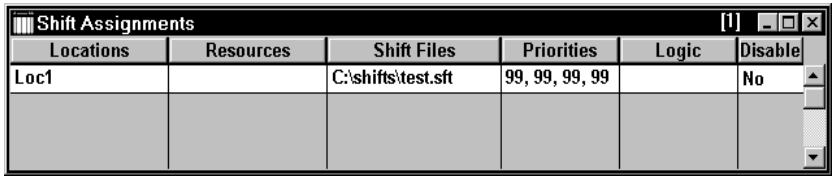


When you select Shifts from the Build menu, two options are displayed: Define and Assign. You must define a shift before you can assign a resource or location to it.

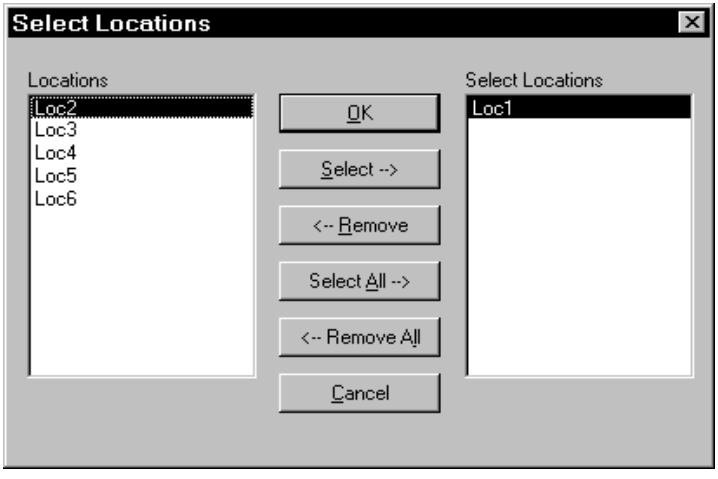
How To

Assign locations and resources to shifts:

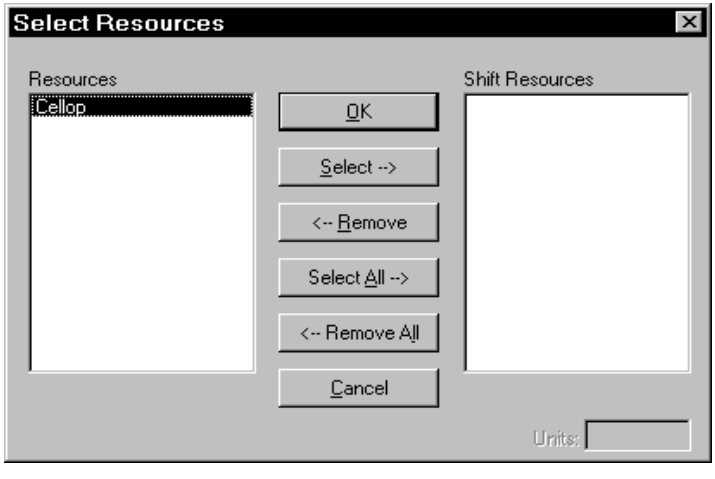
1. Select **Shifts** from the **Build** menu, then click on **Assign**. ProModel displays the Shift Assignments edit window shown below.



2. **Select Locations** - Click on the Locations button to display the Select Locations dialog (*shown below*). Click on a location and use the buttons to select or remove it from the Shift Locations list. Double clicking on a location also selects or removes it. Click **OK** when finished.



3. **Select Resources** - Click on the Resources button to display the Select Resources dialog (*shown below*). Click on a resource and use the buttons to select or remove it from the Shift Resources list. Double clicking on a resource also selects or removes it. Click **OK** when finished.



4. **Units** - Enter the specific units of the selected resource to be assigned to the shift. You may assign one, several, or all units of a resource to a shift. You can also use a macro to specify the units. If left blank, ProModel assigns the default of *All* units to the shift.

   1,3 Units 1 and 3 only

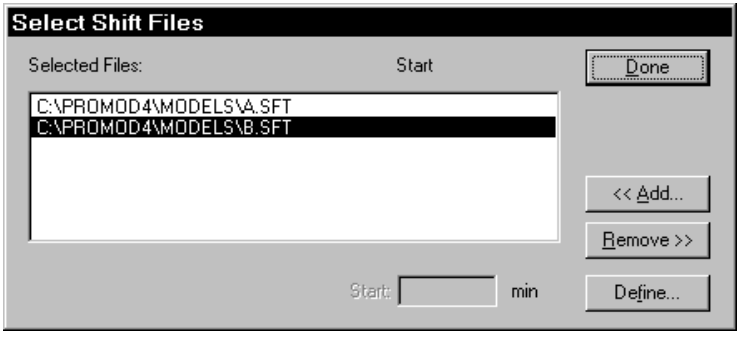
   1-3,5 Units 1 through 3 and 5 only

   All All units of the resource

   none You may use none to indicate that no unit will adopt this shift. This is useful in creating a run-time interface. By using a macro to represent the number of units, the user may select none as an option.

   Macro The name of a run-time interface macro that allows the user to define the units to be affected by the shift.

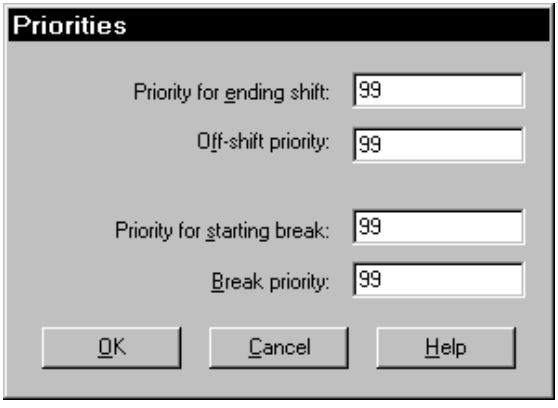
5. **Select Shift Files** - Click the Shift Files button to display the Select Shift Files dialog (*shown next*). Click on the Add button to display the File Open dialog and select the shift files you want to use in the model.



6. **Define Start Times** - Enter the start time for each of the selected shift files. The value will be interpreted according to the time units specified in the General Information dialog unless a unit label is entered after the value (e.g., 10 hr). You can also use a macro to specify the start times. If the Start time is left blank, the shift will begin at the start of the simulation. All shifts specified apply to the locations and resources selected for these shifts in the Shift Assignment record.

   During the simulation, the shift having the earliest start time remains in effect for the locations and resources listed until the next start time encountered activates a new shift.

7. **Define Priorities** - Click on the Priorities button. The Priorities button allows you to enter the priorities for going off-line due to a break or end-of-shift, as well as the priorities of the off-line state in the event that some other activity attempts to bring a particular resource or location back on line. You can also use macros to specify priorities.



These priorities follow standard ProModel priority level and preemption rules. (See *Locations* on page 201, *Entities* on page 237, and *Resources on page 261*.)

Priority for Ending Shift This is the priority for regularly ending the shift. An entity or downtime must have a higher priority level to prevent this location or resource from going off shift at the preset time.

Off Shift Priority This is the priority for the location or resource to stay off shift. In other words, an entity or downtime must have a higher priority level to bring this location or resource back on line before the preset time.

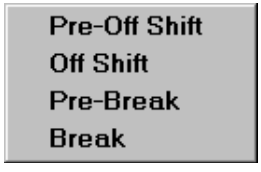
Priority for Starting Break This is the priority for going on break. An entity or downtime must have a higher priority level to prevent this location or resource from going on break at the preset time.

Break Priority This is the priority for staying on break during the break period. In other words, an entity or downtime must have a higher priority level to bring this location or resource back on line before the end of the preset break.

Shift & Break Logic

Shift and break logic are optional and are defined in four distinct logic windows, each executed in a specific sequence throughout the simulation run. You can define logic that controls how resources and locations go off line and what happens once they are off-line.

To define shift or break logic, click on the Logic button to display a submenu of four events associated with shifts for which logic may be defined. Selecting an event from the submenu displays a standard logic window. You can enter separate logic for each of these four events to be executed when the event occurs. See the following discussion, *Sequence of Events.*



You may want to use the Logic Builder to help you enter the logic. Just click on the Build button in the logic window.

Pre-Off-Shift or Pre-Break Logic Executed whenever the location or resource is scheduled to go off shift or on break. This occurs before the location or resource is checked for availability, so it is executed regardless of availability. This logic may be used to check certain conditions before actually taking the resource or location off line. The logic is executed for each resource and location listed as members for this shift assignment record. This allows some members to be taken off line while others may be forced to wait. (Pre-off shift and pre-break logic may be referred to in this manual as pre-logic when speaking of either one.)

Off-Shift & Break Logic Executed at the instant the location or resource actually goes off line.

Sequence of Events

1. When a location or resource is scheduled to go off line due to a break or the end of a shift, the pre-logic for that particular location or resource is executed.
2. After executing the pre-logic, which may contain conditional (WAIT UNTIL) or time (WAIT) delays, the location or resource is taken off-line, assuming it is either available or the priority is high enough for preemption.
3. At the instant the location or resource is taken off line, the Off-Shift or Break logic is executed.

4. After executing this logic, the location or resource waits until the time defined in the shift file expires before going back on line.

Note If the off-shift and break nodes are not specified in the Resource Specs dialog, the resource will stay at the current node. If no resources or locations are assigned to a shift, the shift is ignored.

Functions And Statements

ProModel uses five functions and statements specifically for shift and break logic: SKIP, PRIORITY, DTLENGTH(), FORLOCATION(), and FORRESOURCE(). Following is a brief description of each. For more details, see *Statements & Functions* on page 111 of the *ProModel Reference Guide*.

SKIP If used in pre-logic, it causes the off-shift or break time (including any off-shift or break logic) to be skipped so a location or resource never goes off line. If used in the off-shift or break logic, it causes the off-line time defined in the shift editor to be skipped. This allows you to specify a WAIT statement for the off-line time and SKIP the off-line time defined in the shift editor.

PRIORITY This statement provides an alternative way to specify off-shift or break priorities. It also allows the priority to be changed after some time being off-shift or on break. If the priority is changed to a value lower than the current value, the system will check to see if any preemption may occur at that time. This statement is *not allowed* in off-shift or break pre-start logic.

DTLEFT() This function returns the remaining off-shift time based on when the location or resource is scheduled to go back on shift as defined in the shift file. It may be used in off-shift and break logic to adjust the actual time the location or resource is off-line.

FORLOCATION() This function returns TRUE if the member for which the shift or break logic being executed is a location. This may be followed by a test using the LOCATION() function to determine the precise location.

FORRESOURCE() This function returns TRUE if the member for which the shift or break logic being executed is a resource. The RESOURCE() function may then be used to determine the precise resource if multiple resources are listed as members.

RESOURCE() This returns the name-index number of the resource currently processing the off-shift or break logic.

To illustrate how FORLOCATION() and FORRESOURCE() might be used, consider the following example: Suppose you have locations and resources as members in a shift file assignment and you want to wait until variable **Parts_To_Process** is equal to zero before allowing a particular resource called **Operator** to go off shift. You would enter the following pre-off shift logic:

```
IF FORRESOURCE() THEN
    BEGIN
        IF RESOURCE() = Operator THEN
            BEGIN
            WAIT UNTIL Parts_In_System = 0
            END
    END
```

In addition to these new functions, DTDELAY() may also be called at the beginning of the off-shift or break logic to determine how much time has elapsed between the time the shift downtime was scheduled to start and when it actually started. The length of the shift downtime defined in the shift file would be the sum of DTDELAY() and DTLEFT().

Preemptions to Off-Shift or Break Logic

If off-shift or break logic is defined using WAIT or USE statements and happens to get preempted, the logic will resume one statement after the WAIT or USE statement where it was preempted.

7.7.3 Shift Downtime Principles

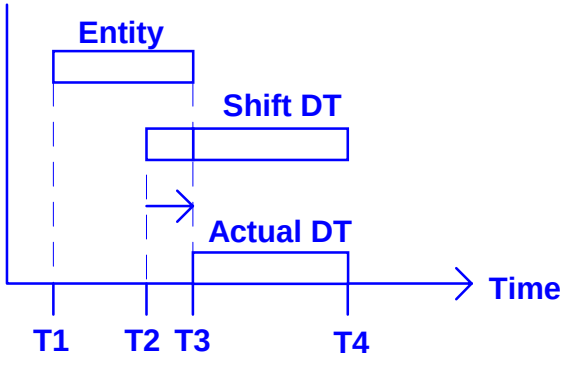
Locations Shift Downtime Principles

It is important to understand that when a location or resource goes off shift, it is essentially down. We call this type of downtime a *shift downtime* and it is treated slightly differently from other downtimes. Breaks, which are also part of the shift schedule, are treated exactly like clock-based downtimes. These downtimes are discussed in *Locations* on page 201 and *Resources on page 261*.

Shift Downtimes for Locations

All location shift downtimes have a default priority of 99, the highest non-preemptive priority possible. This means that when a location is scheduled to go off-shift, this downtime will take priority over all other entities with a priority less than 99 waiting for the location. If the location is currently in use, shift downtimes allow the current entity to complete its process at the location. After the entity is finished, the shift downtime proceeds as if it started at its scheduled time. This means that the location becomes available at the start of the next shift regardless of when it actually went off shift. This procedure is demonstrated in the following example.

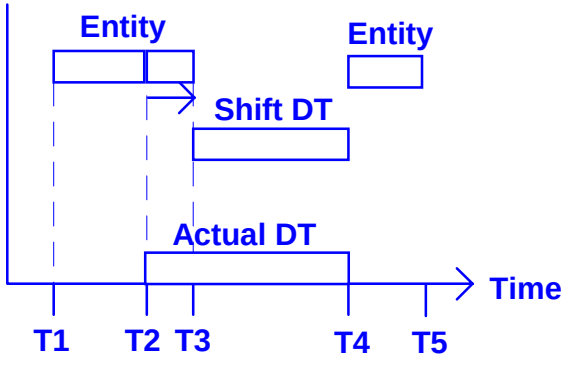
Example 1 (a)



Although the downtime is scheduled to last from time T2 to T4, the actual downtime does not begin until time T3. This is what happens for both locations and resources currently busy when the shift downtime is scheduled to occur.

To preempt a location in which an entity is currently processing, set the priority for going off shift to a number one level higher than the entity's priority.

Example 1 (b)



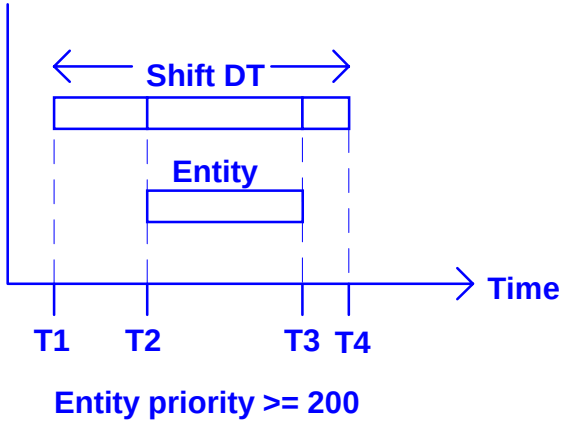
Note Since the entity was preempted, the remaining time for the entity to be processed at the location was completed after the location shift downtime was completed.

Preempting Off-Shift Locations

An off-shift location may be preempted back into service by an entity. Following the preemption, the shift downtime will resume for any remaining time before the start of the next shift. The following example demonstrates this principle.

Example 2

In this example an entity with priority of 200 or greater preempts an off-shift location. The location becomes available to process the entity. Once processing is complete, the location returns to its off-shift status.



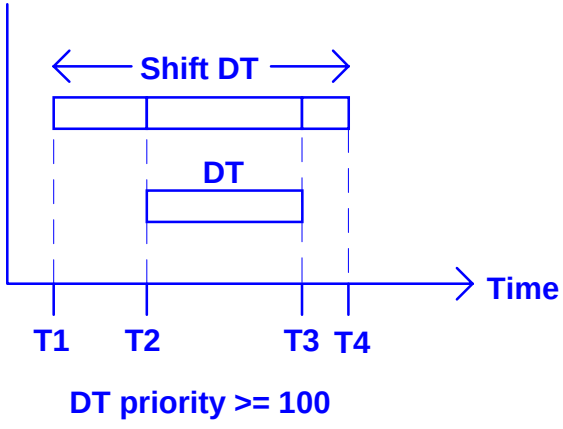
In order for an entity to preempt *any* location downtime (shift or otherwise), it must have a priority level that is at least 2 levels higher than the downtime's priority. In this example the location shift downtime has a priority level of 99 so the entity must have a priority level of 200 or greater to preempt the shift.

Overlapping Downtimes

If a preemptive clock downtime occurs during a shift downtime, the downtimes simply overlap.

Example 3

This example shows the effect of a preemptive downtime occurring for a *location* already off-shift due to a shift downtime. Because location downtimes always overlap, the effect is as if the preemptive downtime never occurred. The location remains off-shift for the total duration of the shift downtime.



The example above could represent the situation where a recurring downtime, such as a lunch or dinner break, has been defined for a single location that is scheduled to be available for a two shift period. It would be simpler to specify a single downtime for lunch and dinner that occurs once every 8 hours continuously than to define separate downtimes for lunch and dinner. In this case the preempting downtime would represent a meal break occurring while the location was off shift.

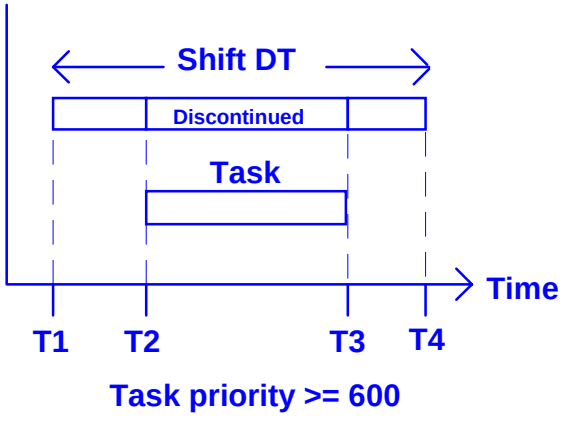
Resource Downtime Principles

Shift Downtimes for Resources

Resource shift downtimes work exactly like location shift downtimes with the exception that if the off-shift downtime is preempted by some other downtime, the original off-shift downtime never resumes. The following examples show how a resource that is off shift is affected by a preemptive request by another entity (example 1) and by downtime preemption (example 2).

Example 1

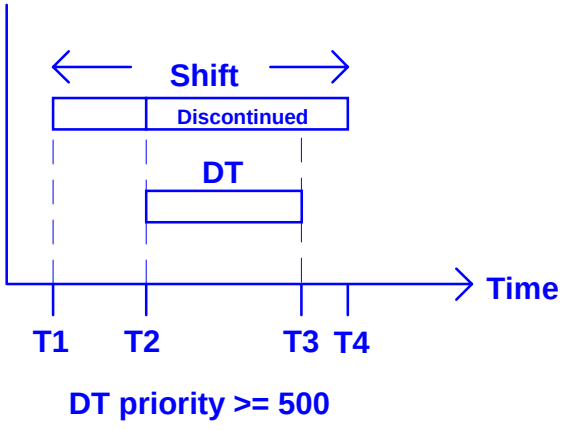
Suppose a resource, repairman, is off-shift. An important machine goes down unexpectedly. Because this machine is a bottleneck in the operation, it is vital to repair the machine as quickly as possible. The repairman is called in and takes 30 minutes to fix the machine. The logic for the downtime to call him back is “USE Repairman, 600 FOR 30 min.” This will preempt the shift downtime and use the repairman to repair the machine even though the repairman is off-shift. Once the repairman has repaired the machine, he returns to his shift downtime until he is scheduled to go back on shift. The repairman's shift downtime will end at the originally scheduled time regardless of the fact it was preempted by a repair activity.



Although the shift downtime is scheduled to last from time T1 to T4, the actual downtime lasts from T1 to T2 and then from T3 to T4.

Example 2

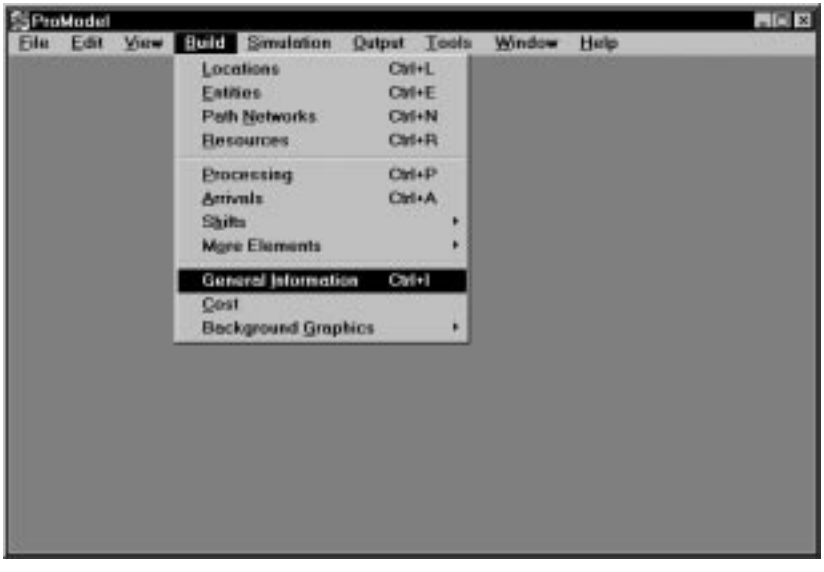
This example shows the effect of a preemptive downtime occurring for a resource already off-shift due to a shift downtime. Since resource downtimes are *not* overlapping, as in the case of location downtimes, the shift downtime in progress is discontinued and the preemptive downtime takes control of the resource because it has a priority greater than or equal to five-hundred (remember that a downtime priority needs to be only one level higher than another downtime priority to preempt it). The effect in this example is that the total downtime is actually shorter than it would have been had the original shift downtime been completed.



Although in practice, situations like the example above are unlikely to occur, it is important to understand that the above condition is possible. Typically, preemptive downtimes are due only to some type of location or resource failure, in which case, the downtime occurrence would be based on usage and not clock time. If a preemptive downtime is based on usage, the situation in the example above could not occur because the location or resource would not be in use, and would not accumulate usage time.

7.8 General Information

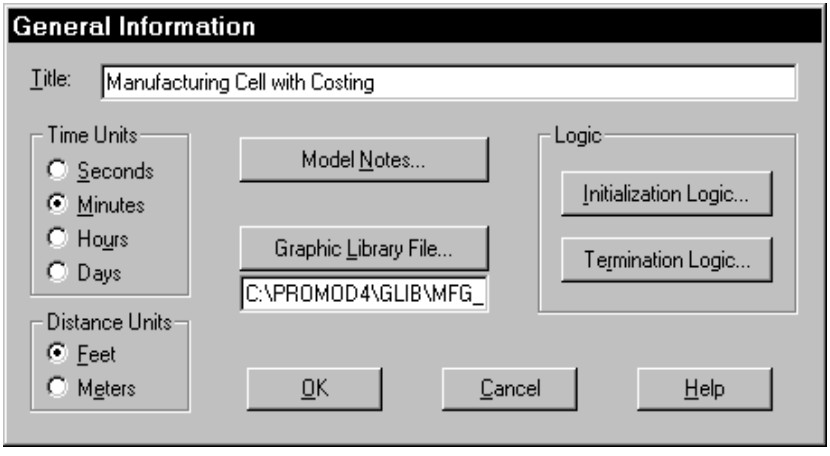
The General Information dialog box allows you to specify basic information about a model, such as its name, default time units, default distance units, and graphic library. You also may specify the model's initialization and termination logic. Finally, a notes window is available for specifying particulars of a model, such as the modeler's name, the revision date, modeling assumptions or anything else about it.



How To **Open the General Information dialog box:**

- Choose **General Information** from the **Build** menu.

7.8.1 General Information Dialog Box



The fields of the General Information dialog box are as follows:

Title An optional, brief description of the model. Information will be displayed in the caption bar and included in the model and results files.

Time Units The unit for any time value in the model that does not have an explicitly specified time unit. The smallest unit of time available in ProModel is .00001 second and the largest is 1 day.

Distance Units The units in feet or meters for all distances specified in the model. There is no practical limit on the size of the model.

Model Notes... Brings up a notes window for specifying general notes about the model. Notes are optional and are for user reference only. An alternative way to display notes is using a DISPLAY statement (see *Display on page 137* of the *ProModel Reference Guide*) in the initialization logic.

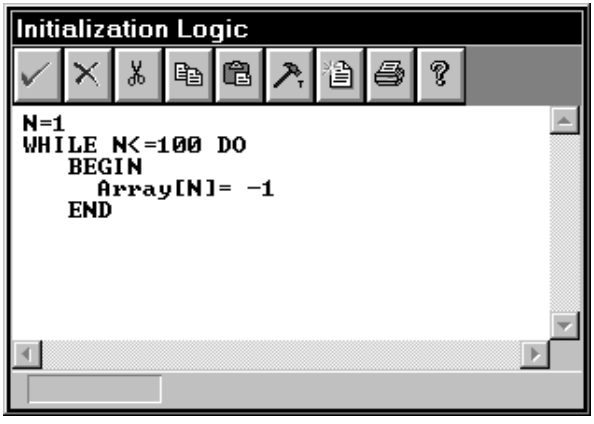
Graphic Library File Opens a dialog box for selecting the graphic library file to use with the open model. Graphics library files have the extension GLB and are further explained later in this section.

Initialization Logic Opens the Initialization Logic window for specifying initialization logic. An asterisk (*) next to the name of the button means that some initialization logic has been defined for the model. (See *Initialization Logic on page 343*.)

Termination Logic Opens the Termination Logic window for specifying termination logic. An asterisk (*) next to the name of the button means that some termination logic has been defined for the model. (See *Termination Logic on page 344*.)

Initialization Logic

Initialization logic allows you to initialize arrays, variables, and other elements at the beginning of a simulation run as shown below:



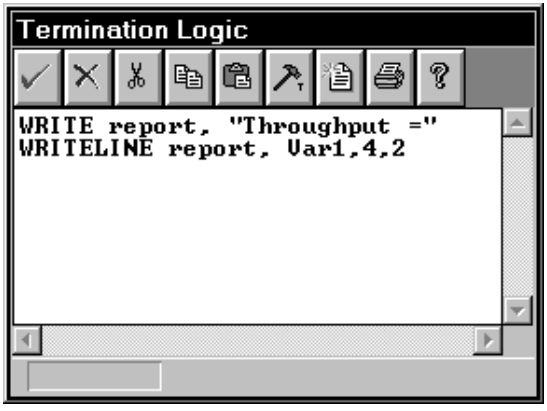
Other common uses of initialization logic include:

- Reading external files
- Displaying messages
- Prompting for values
- Resetting general read/write files
- Activating independent subroutines that process logic based on a timer. (See *Subroutines* on page 447, and the *ProModel Reference Guide*, *Activate* on page 115.)

For a discussion of each of the buttons in the Initialization Logic window, see *Editing Logic Windows* on page 171.

Termination Logic

Termination logic allows you to summarize data or write special statistics to an output file at the end of a simulation run as shown below:



Other common uses of termination logic include:

- Displaying messages
- Resetting read/write files

See the following page for a discussion on the placement of initialization and termination logic within the sequence of run-time events.

Note Although Initialization and Termination logic cannot be tested with the Compile option in the Edit menu, as with Processing or Arrival logic, the logic can be tested by clicking on the compile button in the logic window. When you select OK, ProModel automatically checks all logic and, if it finds an error in the logic, ProModel displays an error message describing the problem.

Execution Time of Initialization and Termination Logic

It is important to understand exactly when initialization and termination logic is executed. When you select Run from the Simulation menu the following things occur in the order listed:

1. Variables are initialized to the values specified in the Variables Editor.
2. Macros with a run-time interface are set to their user-specified value.

3. The model is loaded into the simulation module. As the model is loaded, any numeric expressions used to define such things as location capacities or number of resource units are evaluated and assigned a numeric value.
4. Initialization logic is performed.
5. Simulation begins. Initial arrivals and downtimes are scheduled and simulation processes begin.
6. Simulation ends.
7. Termination logic is performed.
8. Statistics are compiled.

Logic elements that figure into a model's structure are evaluated only when the model is loaded into the simulation module. Those logic elements are:

- Simulation warm-up hours
- Simulation run hours
- Node capacity
- Length of path segments
- Resource units
- Location capacity
- Time and quantity cycle tables
- Queue length
- Conveyor length
- Conveyor speed

For a complete list of when each field is evaluated, see *Appendix A Expressions & Statements* on page 259 of the *ProModel Reference Guide*.

Any variables used in an expression that change any of these logic elements should be initialized in the Variables Editor or run-time interface and not in the initialization logic. The model structure cannot change after the model has been loaded into the simulation module. Thus, any variable figuring into a location's capacity and initialized in the initialization logic will be initialized too late to affect the location's capacity.

Variables which do not figure into a location's capacity may be initialized in the Initialization Logic without any problem. A variable initialized in the initialization logic could be used as the “First Time” for an arrival or downtime occurrence. This is true because arrivals and downtime occurrences are simulation events, and all initialization logic occurs before the first simulation event.

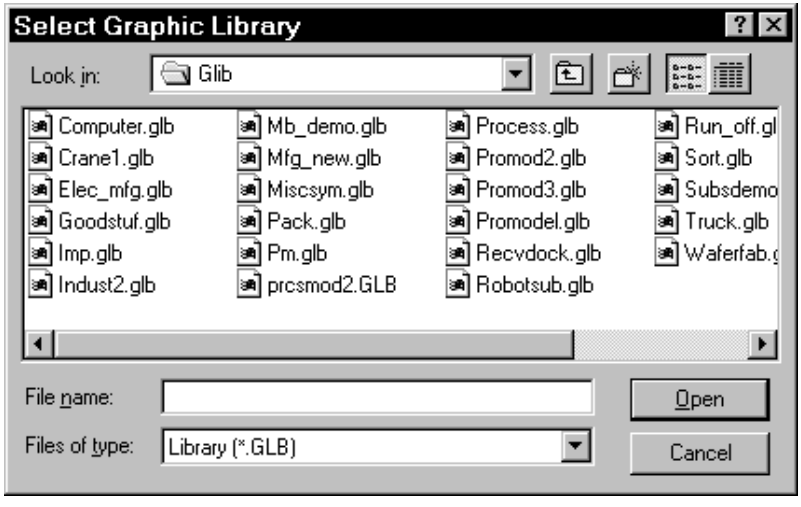
Graphic Library File

ProModel allows you to create and store as many graphics libraries as desired. However, only one graphic library may be used for each model. To copy a graphic from one graphic library to another graphic library, see *Copying a Graphic from One Library to Another* on page 521.

How To

Select the desired graphics library:

1. Select **Graphic Library File** from the General Information dialog box.



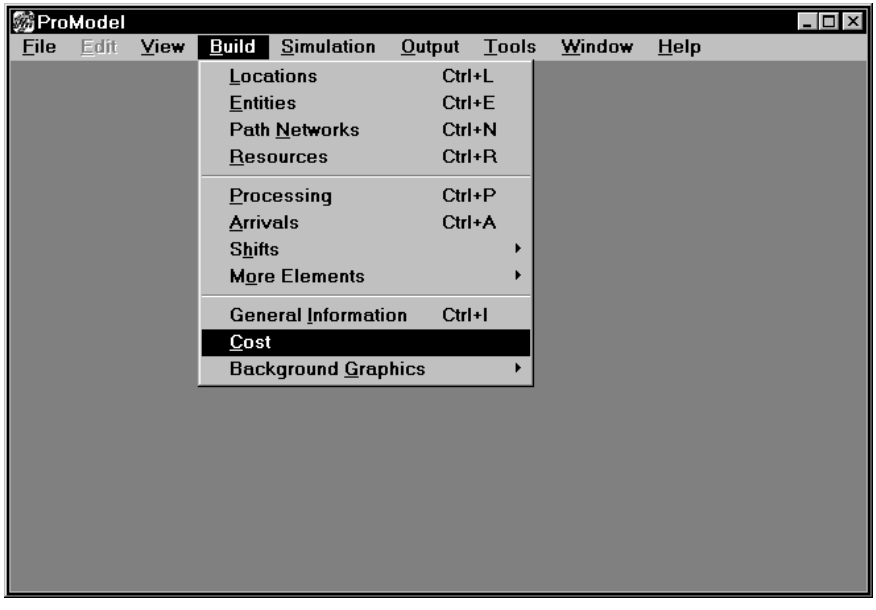
2. Enter the name of the desired graphics library.
3. Select **OK**.

Note

Only files with the extension GLB may be used as graphics libraries. For more information on creating, merging and saving graphics libraries, see *Graphic Editor on page 515*.

7.9 Cost

With ProModel's costing capability, you can make decisions about your system on a cost basis. Costing dialogs allow you to monitor costs associated with Locations, Entities, and Resources during a model run and the General Statistics Report now includes Costing statistics, automatically generated at the start of the simulation.



How To

Use Cost

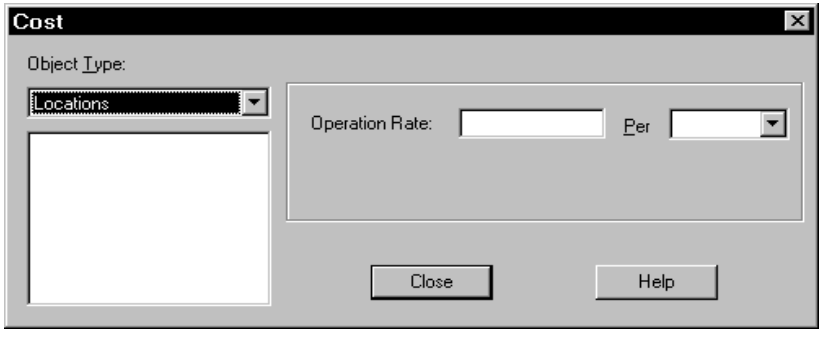
- Choose **Cost** from the **Build** menu. The Cost Dialog appears.

7.9.1 Cost Dialog Box

Use the Cost Dialog box to define costs for Locations, Entities, and Resources. Fields in the Cost Dialog box vary between Object Types and ProModel evaluates expressions in these fields only during translation at run time. The General Statistics Report includes statistical information automatically generated during run-time about cost for locations, entities, and resources.

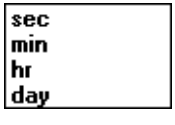
Object Type Use this pull-down menu to define costing for the components of the selected object type. Object types include locations, resources, and entities as shown in the following example. All *defined* model components of the selected type appear in the box below the object type field and ProModel allows you to assign costs to any of these components.

Locations

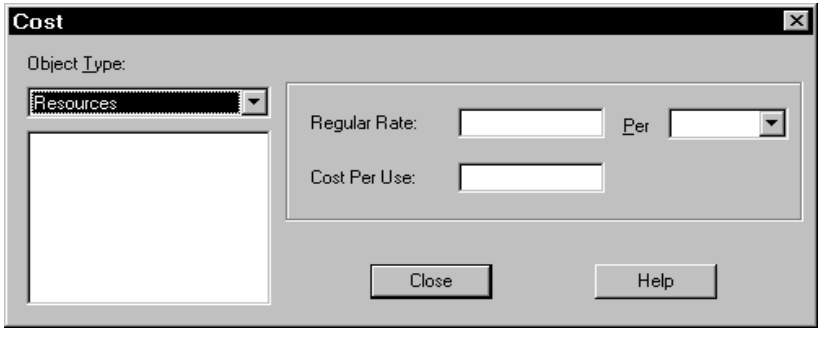


Operation Rate This field specifies the cost per unit of time to process entities at the selected location. Cost accrues only while an entity executes a WAIT or USE statement in operation logic. ProModel accepts expressions in this field and evaluates them at translation.

Per With this pull-down menu, you can set the time units for the Operation Rate. Time units may be in seconds, minutes, hours, or days as shown here.

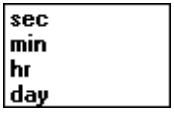


Resources



Regular Rate This field specifies the cost per unit of time for a resource used in the model. You can use expressions in this field (evaluated at translation) to set the rate or change it using SetRate. For more information see *SetRate* on page 225 of the *ProModel Reference Guide*.

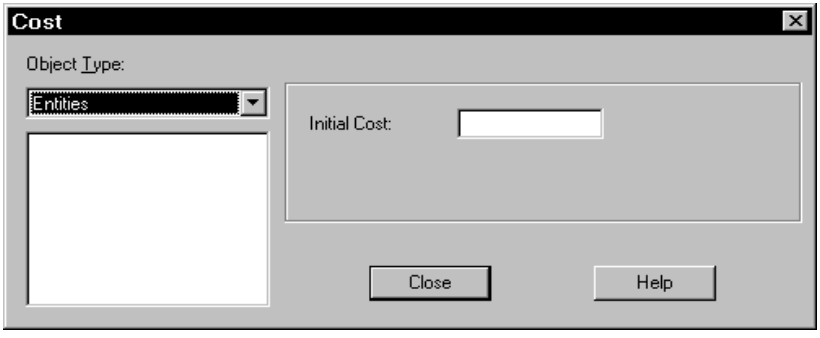
Per This pull-down menu allows you to set the time units for the Regular Rate. Times may be in seconds, minutes, hours, or days as shown here.



Cost Per Use This field allows you to define the actual dollar cost accrued each time you use the resource (i.e., the minimum usage cost). The cost per use updates when you obtain the resource and ProModel accepts expressions in this field (evaluated at translation).

Note Since ProModel counts a preemption as a use, if you preempt a resource from an entity, the usage cost applies to the resource only when it returns to the entity.

Entities

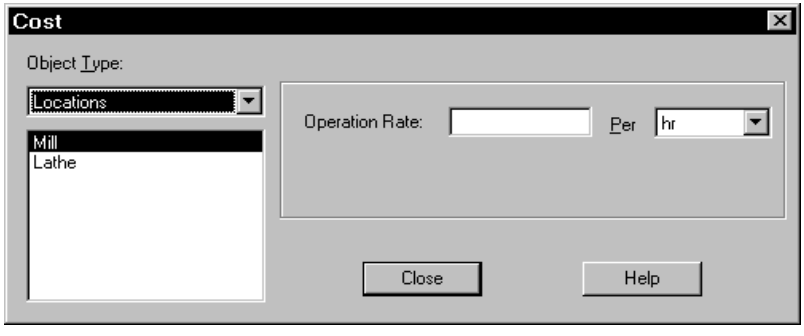


Initial Cost This field allows you to define the initial entity cost for an entity which enters the system through a scheduled arrival, a CREATE statement, or an ORDER statement. ProModel accepts expressions in this field and evaluates them at translation.

Note When you implicitly create new entities through a ROUTE statement, ProModel does not add an initial cost to the entity. To add an initial cost, use the INCENTCOST statement. See *IncEntCost* on page 167 of the *ProModel Reference Guide* for more information.

7.9.2 Building a Model with Costing

When you build a model using the costing feature, you must first define the locations, resources, and entities used in the model. Once you define these model components, you may assign costing information to them through the Cost option in the Build menu. To collect costing information about your model, uncheck the disable costing box from the simulation options dialog of the simulation menu. By default, ProModel disables costing and sets all defaults to zero. See *Enable or Disable Costing* on page 356.



This model shows the impact on product cost and product output when you use a variable number of operators.

Note The following scenarios assume you defined costs for all model components.

7.9.3 Preemption/Downtime

- If you preempt an entity's resource, an additional cost per use will apply once you re-acquire the resource. While waiting for the resource to return, the entity does *not* record operation or resource costs.
- If an entity preempts another entity, the preempted entity continues to record operation time during the entire preemption period. While the preempting and preempted entities are simultaneously at a location, the location records the cost for both entities. If the preempting entity obtains a resource, the preempted entity will *not* record the resource costs during the preemption period.
- If an entity is at a location when a preemptive downtime occurs, the entity records the downtime as part of its operational costs. This applies to all types of location downtimes, including shifts. The location records the cost of the preempted entity while it remains at the location.
- If an entity's resource has a downtime which requires the use of another resource, the entity will *not* record the second resource's cost. However, the location *will* record the extra resource's cost.

7.9.4 Join/Load

- Joined entities add their costs to their base entities, but not their time statistics.
- Loaded entities do *not* add their costs or time statistics to their base entities.
- When an UNLOAD occurs, ProModel divides all costs accrued by a loaded entity among the unloaded entities. ProModel adds all other entity statistics calculated during the loaded period to each of the unloaded entities.
- Entities leaving the system loaded onto other entities do NOT report their individual costs, but *do* report all other statistics. To get the cost of each entity, you must unload the entities before they exit.

7.9.5 Combine/Group

- Combined entities add their costs to the resultant entity, but not their time statistics. The resultant entity begins with fresh time statistics.
- Grouped entities do *not* add their costs or statistics to the group shell (a temporary entity representing grouped entities that starts with cost and time statistics of zero).
- When an UNGROUP occurs, ProModel divides all costs accrued by a grouped entity among the ungrouped entities. ProModel copies all other entity statistics calculated during the grouped period to each of the ungrouped entities.
- Entities leaving the system grouped with other entities do NOT report their individual costs, but *do* report all other statistics. To get the cost of each entity, you must ungroup the entities before they exit.

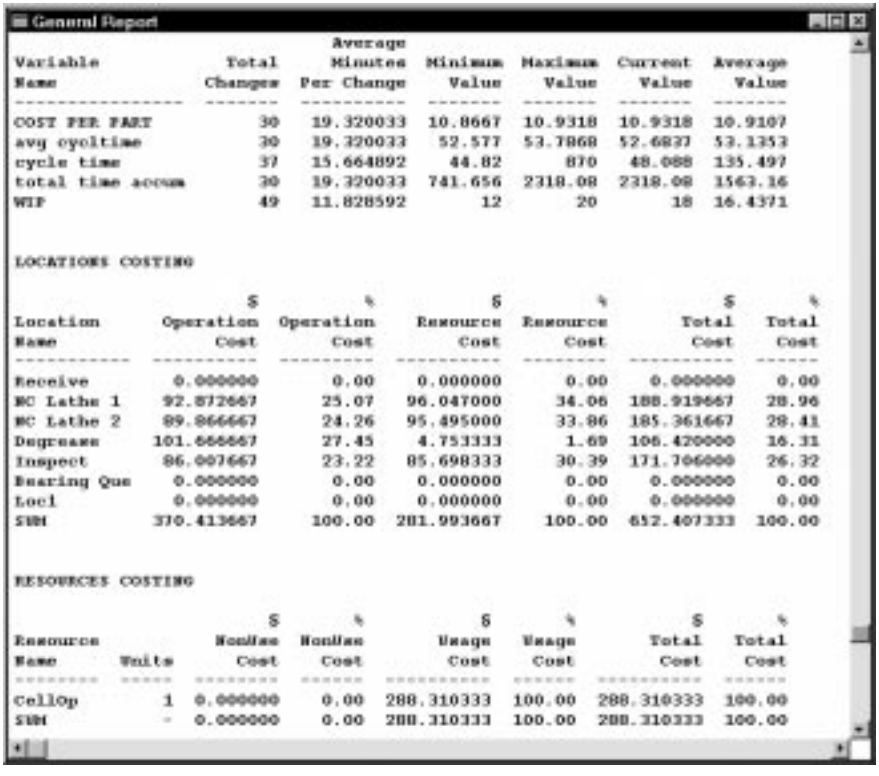
7.9.6 Special Cost Handling

- As soon as you acquire a resource, it begins to accrue cost.
- Unless obtained in the move logic, ProModel charges the “Cost per use” for a resource to the location that obtained it. Resources obtained in the move logic do not charge a “per use” cost to any location.
- ProModel does not charge any resource time used during move logic to any location.
- ProModel adds initial entity costs defined in the cost module only as entity costs, not location costs.
- If a location uses a resource during a downtime, the location accrues that resource’s cost.
- The USE statement counts as operation and resource cost.
- When you CREATE a new entity, it begins with new time statistics and an initial cost.
- If you RENAME an entity, previous time statistics and costs continue with the entity.
- The SPLIT AS statement divides the cost of the original entity between all entities. Each *new* entity begins with new time statistics.

7.9.7 Costing Output

Costing Statistics

ProModel collects costing statistics only if you uncheck the Disable Cost Statistics option in the Simulation Options menu (see *Simulation Options* on page 569 for more information). Included in the General Statistics Report, ProModel calculates costing statistics as shown in the following example:



General Report

Variable Name	Total Changes	Average Minutes Per Change	Minimum Value	Maximum Value	Current Value	Average Value
COST PER PART	30	19.320033	10.8667	10.9318	10.9318	10.9107
avg cycltime	30	19.320033	52.577	53.7868	52.6837	53.1353
cycle time	37	15.664892	44.82	870	48.088	135.497
total time accum	30	19.320033	741.656	2318.08	2318.08	1563.16
WIP	49	11.828592	12	20	18	16.4371

LOCATIONS COSTING

Location Name	$ Operation Cost	% Operation Cost	$ Resource Cost	% Resource Cost	$ Total Cost	% Total Cost
Receive	0.000000	0.00	0.000000	0.00	0.000000	0.00
MC Lathe 1	92.872667	25.07	96.047000	34.06	188.919667	28.96
MC Lathe 2	89.866667	24.26	95.495000	33.86	185.361667	28.41
Degrease	101.666667	27.45	4.753333	1.69	106.420000	16.31
Inspect	86.007667	23.22	85.698333	30.39	171.706000	26.32
Bearing Que	0.000000	0.00	0.000000	0.00	0.000000	0.00
Loc1	0.000000	0.00	0.000000	0.00	0.000000	0.00
SUM	370.413667	100.00	281.993667	100.00	652.407333	100.00

RESOURCES COSTING

Resource Name	Units	$ NonUse Cost	% NonUse Cost	$ Usage Cost	% Usage Cost	$ Total Cost	% Total Cost
CellOp	1	0.000000	0.00	288.310333	100.00	288.310333	100.00
SUM	-	0.000000	0.00	288.310333	100.00	288.310333	100.00

Locations

- Operational Cost = (Active Operation Time * Rate) + (Any IncLocCost)
- % Operational Cost refers to the location's percentage of the sum of all operation costs
- Resource Cost = (Utilization * Rate) + (Times Used * Cost per use)

Note For Resource Cost, Utilization and Times Used refer to the utilization of a resource while at a location. This applies only to resource use through operation logic.

- % Resource Cost refers to the location's percentage of the sum of all resource costs
- Total Cost = (Operation Cost + Resource Cost)
- % Total Cost refers to location's percentage of the sum of all location costs

Resources

- NonUse Cost = (1-% Utilization) * Scheduled Time * Rate
- % NonUse Cost refers to the resource's percentage of the sum of all nonuse costs
- Usage Cost = (% Utilization * Scheduled Time * Rate) + (Times Used * Cost per use)
- % Usage Cost refers to the resource's percentage of the sum of all resource usage costs
- Total Cost = Usage Cost + NonUse Cost
- % Total Cost refers to the resource's percentage of the sum of all resource costs

Entities

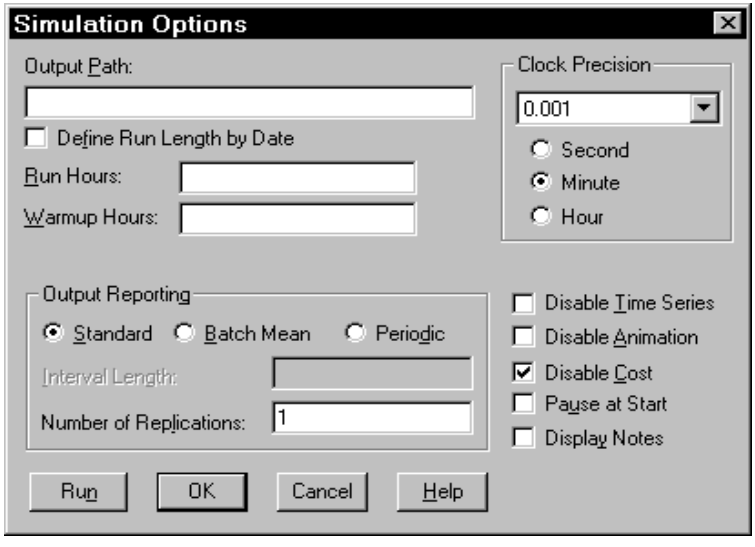
- Total Cost = cumulative entity cost, or the sum of costs incurred on all locations the entity passed through + the sum of all costs incurred by use of resource + initial cost + any IncEntCost
- % Total Cost refers to the entity's percentage of sum of all entity costs

In the above calculations, the rate defined (per day, hour, minute, and second) converts to the default time units specified in the General Information dialog.

Note ProModel does not allow you to generate a Costing Graph. However, if you set a variable equal to GetCost (e.g., Var1=GetCost), you can generate a time series graph to track changing entity costs. See *GetCost()* on page 157 of the *ProModel Reference Guide* for more information.

7.9.8 Enable or Disable Costing

By default, ProModel automatically *disables* the Costing option. To *enable* the costing feature, uncheck the Display Cost Statistics option in the Simulation Options dialog.



How To Enable Costing

1. From the **Simulation** menu, select **Options**.
2. *Uncheck* the **Disable Cost** box.

How To Disable Costing

1. From the **Simulation** menu, select **Options**.
2. Check the **Disable Cost** box.

7.10 Tanks

Tanks are simply locations to which ProModel associates a level instead of an entity routing. (As a result, the units and rules fields do not apply to tanks.) Using tanks, you can model the continuous flow of liquids and other substances into and out of tanks or similar vessels. Also, when combined with discrete-event simulation, ProModel's continuous modeling capability makes it possible to model the exchange between continuous material and discrete entities (e.g., when you place liquid into a container). Other uses include modeling high-rate, discrete part manufacturing systems.

The Tank Submodel

In order to function properly, all tank models should include the tank submodel (TANKSUB.MOD). The tank submodel contains important subroutines and data elements (arrays and macros) used to simplify tank modeling. Each of these subroutines and data elements has a "Tank_" prefix to help identify it and to prevent any accidental name duplication.

Note All user-defined model elements should begin with something other than "Tank_".

How To

Define a tank

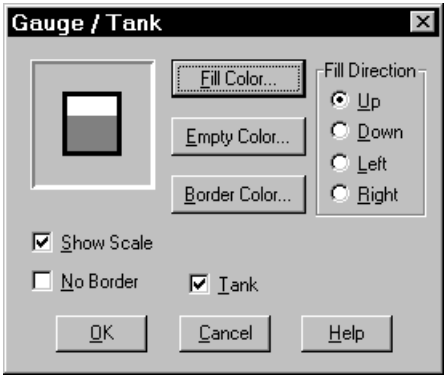
1. Select the gauge/tank symbol from the Location Graphics window.
2. Click on the layout window where you wish to place the tank and select **Create Tank Location** from the menu that appears. ProModel places the tank on the layout.

Note When you create the first tank in your model, ProModel will display a dialog that allows you to automatically import various subroutines, arrays, macros, and library graphics specific to tanks. If you do not wish to include these new items, you may cancel the action.

3. Enter a capacity (1 to 999999) for the tank in the location capacity field.
4. Define and reference any necessary tank control subroutines.

How To Edit a tank or a gauge

1. Double click on the tank or gauge (or right click and select **Edit Graphic**).
2. From the dialog that appears, make the appropriate changes.
3. Click **OK**.



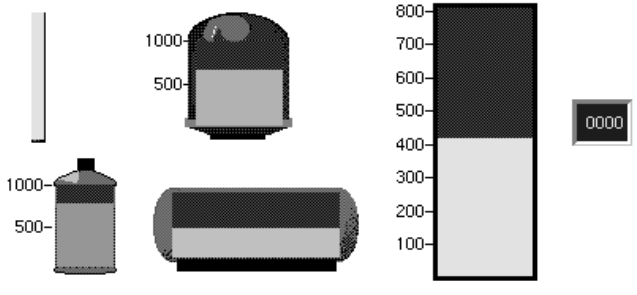
How To Change between a tank and a gauge

- Double click on the tank or gauge and check or uncheck the tank option.

or...

- Right click on the tank or gauge and select **Change Tank to Gauge** or **Change Gauge to Tank**.

In addition to defining a tank graphic, you may add labels and other figures to a tank. For example, you can add a counter to digitally display the fill level of the tank (ProModel rounds the value displayed to the nearest integer). The following are examples of how you can use tanks in ProModel.



7.10.1 Basic Concepts

Since tanks do not process discrete entities, you may not define routings to or from tanks. To control a tank level, ProModel provides predefined subroutines that fill, empty, and transfer tank contents. To monitor tank levels and initiate flows, you must define control subroutines using the Subroutine module. To call these subroutines and operate them independently in the model, use the ACTIVATE statement. For examples of how to use these subroutines, see the discussion at the end of this section. To model tanks effectively, you must understand the following concepts.

Tank Levels

ProModel records tank levels in a pre-defined array called Tank_Level where each element of the array corresponds to each tank location in the location list. For example, the level of TankA is the value of Tank_Level [TankA]. If TankA were the third location in the location list, you could also reference the level of TankA with Tank_Level[3]. For best results, you should control all tank levels using only the predefined tank filling and emptying subroutines rather than change the Tank_Level array values directly. This will prevent overfilling or overdrawing and will accurately gather statistics for each tank. For example, calling Tank_Fill (TankA, 500, 30, 0) automatically fills TankA to 500 units at a rate of 30 units per minute. The 0 signifies that the tank will not accept excess material and, as a result, an error message will occur if the tank reaches capacity before the specified amount fills into the tank.

The Flow Time Step

To model continuous flow, ProModel uses a Tank_TimeStep macro. This macro is the time step used when filling/emptying tanks and is an RTI (run-time interface) parameter. Initially, ProModel sets this value to .2 minutes. If you wish to use a different value for the time step, you may change it *temporarily* (for a particular model) through the Simulation/Parameters menu option, or *permanently* by changing the macro itself. The larger the time step, the longer the interval between filling and emptying (which speeds up the simulation). For example, suppose you set the time step to .1 minutes. If a tank empties at a rate of 60 gpm, the simulation would actually empty the tank by a discrete amount of 6 gallons every .1 minutes. When filling or emptying a tank, if the remaining quantity doesn't require the full time step, ProModel reduces the time step using a linear interpolation.

Note The only adverse effect of using a large time step is that any WAIT UNTIL statement or other test based on the Tank_Level array may be off by as much as the flow amount for the time step. For example, if the time step is .5 minutes and the rate of flow is 60 gpm, the level will change in 30 gallon increments. This means that the tank will

not satisfy the statement “WAIT UNTIL Tank_Level[TankA]>=31” until the level reaches 60.

Rate of Flow

To use flow rates properly, you must define all rates in terms of units (i.e., gallons, pounds, etc.) per minute. Whenever you call one of the empty, fill, or transfer subroutines, you must specify the rate of flow. The units of flow, however, may change when you move material from one tank to another (e.g., pounds of dry material may transfer into a tank containing gallons of liquid).

To specify a variable rate of flow that changes dynamically with each time step, pass a value of 0. This signals the subroutine to call the Tank_Rate subroutine with each time step. To return the desired rate value for each time step when you use a variable rate, you must modify the Tank_Rate subroutine appropriately.

Tank States

Like other model elements, tanks use states to test and track statistics. ProModel automatically sets these states when you use the predefined tank subroutines to control the tank. The following are defined states:

Tank_Idle The tank is empty and not in use. Set automatically when a tank empties and at the end of a Tank_DoPrep or Tank_GoDown subroutine.

Tank_Operation The tank is active (e.g., mixing, reacting, heating, etc.). Set automatically when the model calls the Tank_DoOperation subroutine.

Tank_Setup The tank is cleaning or preparing for future use. Set automatically whenever you call the Tank_Prep subroutine.

Tank_Filling The tank is filling. Set automatically whenever you fill the tank.

Tank_Emptying The tank is emptying. Set automatically whenever you empty the tank.

Tank_Blocked The tank is full and ready to transfer. Set automatically when the tank fills to capacity.

Tank_Down The tank is down. Set automatically whenever you call Tank_GoDown.

While ProModel sets these states automatically, you may change the state of the tank at any time by calling the Tank_SetState subroutine. ProModel records statistics for these states in the output report under Locations. Since a tank may fill and empty simultaneously, the output report combines Tank_Filling with Tank_Emptying and reports it all as waiting time.

Over Filling/Emptying Tanks

When using the predefined subroutines to fill, empty, or transfer from one tank to another, you may accidentally attempt over fill or over empty a tank. To prevent these situations, you have the option to terminate the fill/empty subroutine or suspend further filling/emptying until the tank reaches a resume level. If you terminate the subroutine, ProModel temporarily stores the un-filled or un-emptied quantity for immediate access in the global variable, Tank_QtyLeft.

Tank Downtimes

For Tanks, you must define downtimes and shifts in a special way. First, you may define only clock downtimes for tanks. Second, when defining a clock downtime for a tank, use the Tank_GoDown subroutine (page 377) in the Downtime Logic field instead of just a WAIT statement. This sets the state of the tank to Tank_Down and gathers the appropriate statistics. Third, when defining a shift for a tank, you should call the Tank_GoDown subroutine in the off-shift logic using the DTLeft() function as the time parameter. A SKIP statement should follow this function as shown here:

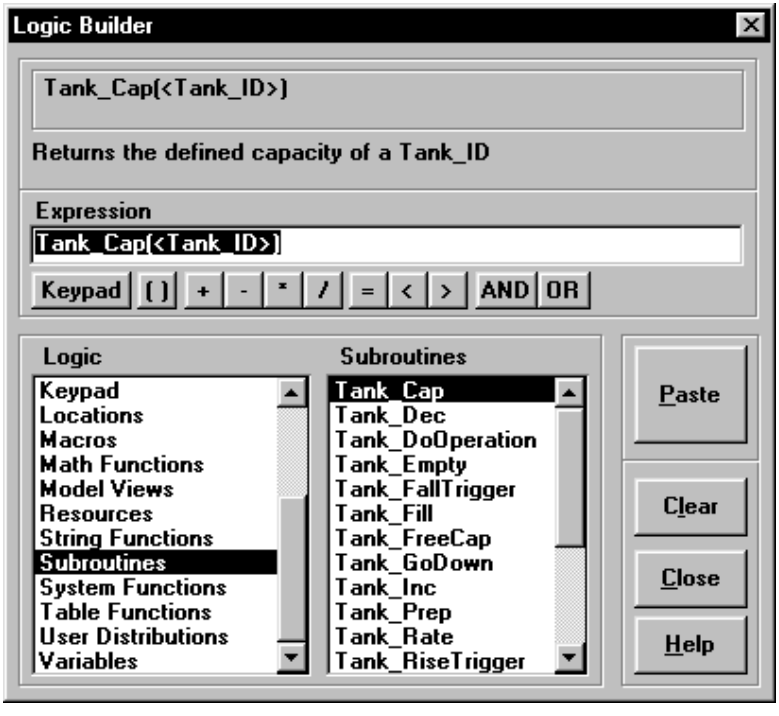
```
Tank_GoDown (<TankID>, DTLeft())
SKIP
```

ProModel temporarily suspends tank flow while a tank is down or off shift.

7.10.2 Tank Logic Builder

An expanded capability within ProModel, the *tank* logic builder provides you with what you need to model complex tank and fluid system operations. The logic builder contains all available tank subroutines and provides you with a description of the components required to use each subroutine.

Note The subroutine logic is not accessible until you define your first tank location—when you define the tank location, ProModel loads the tank submodel.



How To

Access the Logic Builder:

- Click the right mouse button in the logic window or expression edit field. Or click the **Build** button on the logic window's toolbar.

For more information about the Logic Builder, see *Logic Builder on page 489*.

7.10.3 Pre-defined Tank Subroutines

Tank_Fill

Syntax TANK_FILL (*<Tank ID>, <Fill Quantity>, <Fill Rate>, <Resume Level>*)
TANK_FILL (HoldingTank, 2000, 75, 1500)

Description Fills a tank using a specific quantity and rate. The default tank state sets to Tank_Filling, then to Tank_Blocked if the tank becomes full.

Use Tank_Fill when the source of the material is not another tank, but an arriving entity or a source that is not part of the model.

Components

<Tank ID> The tank name or location index number.

<Fill Quantity> The number of units (gallons, pounds, etc.) to fill into the tank. To fill the tank to capacity, enter Tank_Cap(<Tank ID>).

<Fill Rate> The rate in units (gallons, pounds, etc.) per minute. To instantly increase the level of a tank, use the Tank_Inc subroutine. To initialize the level of a tank (e.g., at the start of the simulation), use the Tank_SetLevel subroutine. To use a dynamically calculated rate in the Tank_Rate subroutine, enter 0.

<Resume level> If the tank level reaches capacity before you add the specified quantity, the tank must drop to the resume level before it can continue filling. To terminate filling if the tank reaches capacity, enter Tank_Stop as the resume level. A value of 0 causes an error to occur if the tank becomes full before reaching the fill quantity.

☞ **Example** A tanker arrives and fills a storage tank by the quantity stored in the tanker's attribute, Load_Qty. The rate of fill is 80 gpm and, if the tank fills to capacity before the tanker discharges the entire quantity, the level of the storage tank must drop to 12,000 gallons before it resumes filling. To represent this, enter the following statement in the operation logic for the tanker at the unloading station:

```
Tank_Fill(StorageTank, Load_Qty, 80, 12000)
```

See Also *Filling from an Entity* on page 390 and *Initializing and Replenishing Supply Tanks* on page 390.

Tank_Empty

Syntax TANK_EMPTY (<*Tank ID*>, <*Empty Quantity*>, <*Empty Rate*>, <*Resume Level*>)
TANK_EMPTY (TankB, 2000, 40, 0)

Description Empties a tank by a specified quantity and rate. The state is set to Tank_Emptying, then to Tank_Idle if the tank becomes empty.

Use Tank_Empty when the destination is not another tank, but an arriving entity or a source that is not part of the model.

Components

<*Tank ID*> The tank name or location index number.

<*Empty Quantity*> The number of units (gallons, pounds, etc.) to empty. To empty a tank completely of its current contents, enter Tank_Level [<Tank ID>].

<*Empty Rate*> The rate in units (gallons, pounds, etc.) per minute. To instantly decrease the level of a tank, use the Tank_Dec subroutine. To specify a dynamically calculated rate using the Tank_Rate subroutine, enter 0.

<*Resume level*> If the tank level drops to 0 before you empty the specified quantity, the tank must rise to the resume level before continuing to empty. To terminate emptying if the level ever drops to 0, enter Tank_Stop. A value of 0 causes an error to occur if the tank becomes empty before removing the specified quantity.

☞ **Example** When a chemical tank, ChemTank, is full (state is Tank_Blocked), workers pump its contents into a rail car at a rate of 60 gpm for transportation to another facility. Since rail cars are always available and the delivery activity is not of interest, it is not necessary to model the rail cars explicitly. Instead, activate a subroutine in the initialization logic with the following statement:

```
Tank_Loop //causes logic to repeat continuously
    {
        WAIT UNTIL Tank_State[ChemTank]=Tank_Blocked
        Tank_Fill(ChemTank, Tank_Level[ChemTank], 60, 0)
    }
```

See Also *Emptying to an Entity* on page 392.

Tank_Transfer

Syntax TANK_TRANSFER (<*FROM Tank ID*>, <*TO Tank ID*>, <*Transfer Quantity*>, <*FROM Rate*>, <*TO Rate*>, <*Resume Level*>)
TANK_TRANSFER (Tank1, Tank2, 2000, 100, 0, 0)

Description Transfers a specified quantity from one tank to another. ProModel sets the state of the FROM tank to Tank_Emptying and the TO tank to Tank_Filling. If the FROM tank becomes empty, its state becomes Tank_Idle. If the TO tank becomes full, its state becomes Tank_Blocked. Otherwise, the states remain unchanged.

Use Tank_Transfer when you want to transfer a specific quantity from one tank to another.

Components

<*FROM Tank ID*>The name or location index number of the FROM tank.

<*TO Tank ID*> The name or location index number of the TO tank.

<*Transfer Quantity*>The number of units (gallons, pounds, etc.) to transfer. To transfer the entire contents of a tank, enter Tank_Level [<FROM Tank ID>].

<*FROM Rate*> The rate in units (gallons, pounds, etc.) per minute out of the FROM tank. To use a dynamically calculated rate in the Tank_Rate subroutine, enter 0.

<*TO Rate*> The rate in units (gallons, pounds, etc.) per minute into the TO tank. Use 0 if same as the FROM rate. (The TO rate is automatically the same as the FROM rate if you add 0 as the FROM rate.)

<*Resume level*>If the TO tank reaches capacity before the specified quantity transfers, the TO tank must drop to the resume level before continuing with the transfer. To terminate transferring when the TO tank reaches capacity, enter Tank_Stop. A value of 0 causes an error to occur if the tank becomes empty before transferring the specified quantity.

☞ **Example** When a mixing tank is ready to mix a new batch of material, 10,000 gallons of water must first transfer from a water supply tank at a rate of 100 gpm. The following logic represents this action:

```
Tank_Transfer(WaterTank, MixingTank, 10000, 100, 0, 0)
```

See Also *Tank Transfers* on page 393.

Tank_TransferUpTo

Syntax TANK_TRANSFERUPTO (*<FROM Tank ID>, <TO Tank ID>, <TO Level >, <FROM Rate>, <TO Rate>*)
TANK_TRANSFERUPTO (Tank1, Tank2, 8500, 75, 0)

Description Similar to Tank_Transfer except that Tank_TransferUpTo does NOT terminate a transfer based on the transferred *quantity*, rather when the TO tank *level* rises to a certain point. If the tank empties before reaching the TO level, ProModel suspends the transfer until capacity becomes available.

Use Tank_TransferUpTo when you want to raise the level of a tank to a certain value but are not certain of the quantity needed to reach that level (e.g., the tank is draining at the same time you are trying to fill it).

Components

<FROM Tank ID> The name or location index number of the FROM tank.

<TO Tank ID> The name or location index number of the TO tank.

<TO Level> Transfer until the TO tank reaches this level.

<FROM Rate> The rate in units (gallons, pounds, etc.) per minute out of the FROM tank. To use a dynamically calculated rate in the Tank_Rate subroutine, enter 0.

<TO Rate> The rate in units (gallons, pounds, etc.) per minute into the TO tank. Use 0 if the TO rate is the same as the FROM rate.

☞ **Example** An in-process tank supplies several downstream tanks and must maintain a maximum level of 20,000 gallons. Whenever the in-process tank drops below 5,000 gallons, a supply tank refills the tank at a rate of 100 gpm. To model this, define an activated subroutine for the supply tank using the following logic:

```
Tank_Loop //causes logic to repeat continuously
  {
  WAIT UNTIL Tank_Level[InProcessTank]<=5000
  Tank_TransferUpTo(SupplyTank, InProcessTank, 20000, 100, 0)
  }
```

Tank_TransferDownTo

Syntax TANK_TRANSFERDOWNTO (<*FROM Tank ID*>, <*TO Tank ID*>, <*TO Level* >, <*FROM Rate*>, <*TO Rate*>)
TANK_TRANSFERDOWNTO (Tank1, Tank2, 1000, 80, 0)

Description Similar to Tank_Transfer except that Tank_TransferDownTo terminates the transfer when the FROM tank level lowers to a designated level instead of lowering by a specific quantity. If the TO tank becomes full, ProModel suspends the transfer until capacity becomes available.

Use Tank_TransferDownTo when you want to lower the level of a tank to a specific value but you are not certain how much to empty in order to drop to that level (e.g., the tank may fill at the same time it empties).

Components

<*FROM Tank ID*>The name or location index number of the FROM tank.

<*TO Tank ID*> The name or location index number of the TO tank.

<*TO Level*> Transfer until the FROM tank drops to this level.

<*FROM Rate*> The rate in units (gallons, pounds, etc.) per minute out of the FROM tank. To use a dynamically calculated rate in the Tank_Rate subroutine, enter 0.

<*TO Rate*> The rate in units (gallons, pounds, etc.) per minute into the TO tank. Use 0 if the TO rate is the same as the FROM rate.

☞ **Example** An in-process tank, TankA, supplies TankB at a rate of 50 gpm. TankA must maintain a minimum level of 200 gallons to insure against pump cavitation. When TankA's level drops to 200 gallons, the tank stops pumping to TankB until the level of TankA rises above 200 gallons. To model this scenario, enter the following logic in the subroutine controlling the flow from TankA to TankB:

```
Tank_Loop //causes the logic to repeat continuously
  {
  WAIT UNTIL Tank_Level[TankA]>200
  Tank_TransferDownTo(TankA, TankB, 200, 50, 0)
  }
```

See Also *Split Transfers* on page 395.

Tank_SetLevel

Syntax TANK_SETLEVEL (<*Tank ID*>, <*Quantity*>)
TANK_SETLEVEL (TankA, 1500)

Description Instantly sets the level of a tank to a specified quantity. If the quantity is negative or larger than the tank capacity, an error occurs. The tank state sets to Tank_Blocked if you set the tank level to the tank capacity and to Tank_Idle if you set the tank level to 0. Otherwise, the state remains unchanged.

Use Tank_SetLevel when you want to initialize a tank to a specific level.

Components

<*Tank ID*> The tank name or location index number.

<*Quantity*> The level at which to set the tank (number of gallons, pounds, etc.). To completely fill the tank, enter Tank_Cap(<Tank Name>).

☞ **Example** When you begin a simulation, you wish to set the initial level of a supply tank, TankX, to 10,000 gallons. To model this, enter the following statement in the initialization logic for the model.

```
Tank_SetLevel(TankX, 10000)
```

See Also *Initializing and Replenishing Supply Tanks* on page 390.

Tank_Inc

Syntax TANK_INC (<*Tank ID*>, <*Quantity*>)
TANK_INC (StorageTank, 5000)

Description Instantly increases the level of a tank by a specified quantity. If the tank has insufficient capacity, the level increases as capacity becomes available. ProModel sets the tank state to Tank_Blocked if the level increases to the tank capacity, otherwise the state remains unchanged.

Use Tank_Inc to instantly add a specific quantity to a tank.

Components

<*Tank ID*>	The tank name or location index number.
<*Quantity*>	The number of units by which to increment the contents of the tank (gallons, pounds, etc.).

☞ **Example** Trucks deliver pellets to a holding bin twice a day. When a truck arrives at the drop-off station, it dumps the entire 5,000 lb load in only 2.5 minutes. To model this, define the following operation logic for the truck at the drop-off station:

```
WAIT 2.5 MIN
Tank_Inc(HoldingBin, 5000)
```

Tank_Dec

Syntax TANK_DEC (<*Tank ID*>, <*Quantity*>)
TANK_DEC (SupplyTankB, 1000)

Description Instantly decreases the level of a tank by a specified quantity. If the tank has insufficient quantity, it empties as material becomes available. ProModel sets the tank state to Tank_Idle if you decrease the level to 0. Otherwise the state remains unchanged.

Use Tank_Dec to instantly remove a specific quantity from a tank.

Components

<*Tank ID*> The tank name or location index number.

<*Quantity*> The number of units by which to decrement the contents of the tank (gallons, pounds, etc.).

☞ **Example** A fill tank fills one 10-gallon container every 15 seconds. After filling, each container moves to a location called FillStation. To model this activity, define the following activated subroutine (this subroutine creates a filled container every 15 seconds):

```
Tank_Loop //causes logic to repeat continuously
  {
  WAIT 15 SEC
  Tank_Dec(FillTank, 10)
  ORDER 1 Container TO FillStation
  }
```

See Also *Emptying to an Entity* on page 392.

Tank_RiseTrigger

Syntax TANK_RISETRIGGER (<*Tank ID*>, <*Level*>)
TANK_RISETRIGGER (TankA, 3000)

Description Waits until tank contents rises to a specific level.

Use Tank_RiseTrigger to initiate some action when a tank rises to a certain level.

Components

<*Tank ID*>	The tank name or location index number.
<*Level*>	When the tank level rises to this value, ProModel executes any subsequent logic.

Example A tanker waits at a dispatch station until the level of a finished goods tank rises to 2,000 gallons. Once the tank level reaches this point, a signal dispatches the tanker to the finished goods tank for loading. Meanwhile, the finished goods tank continues filling. To model this situation, define the following process logic for the tanker at the dispatch station:

```
Tank_RiseTrigger (FGTank, 2000)
```

Note Using the Tank_RiseTrigger subroutine instead of a WAIT UNTIL statement prevents the next tanker from dispatching until the finished goods tank falls back below 2,000 gallons.

See Also *Defining Trigger Levels* on page 397.

Tank_FallTrigger

Syntax TANK_FALLTRIGGER (<*Tank ID*>, <*Level*>)
TANK_FALLTRIGGER (TankB, 500)

Description Waits until tank contents falls to a specified level.

Use Tank_FallTrigger to initiate an action when a tank level falls to a specific level.

Components

<*Tank ID*>	The tank name or location index number.
<*Level*>	When the tank level falls to this value, any subsequent logic executes.

☞ **Example** When an in-process tank, TankX, falls to 1000 gallons, it triggers a mixing tank to begin producing more product. To model this, define the following activated subroutine to control the mixing tank:

```
Tank_Loop //causes the logic to repeat continuously
  {
  Tank_FallTrigger(TankX, 1000)
  [Insert logic to mix new batch here]
  }
```

Note Using Tank_FallTrigger instead of a WAIT UNTIL statement prevents the action from triggering again until the level first rises above the fall trigger level.

See Also *Defining Trigger Levels* on page 397

Tank_Cap

Syntax TANK_CAP (<*Tank ID*>)
TANK_CAP (TankA)

Description Returns the capacity defined for the specified tank.

Use Tank_Cap when you need to know the defined capacity for a tank.

Components

<*Tank ID*> The tank name or location index number.

Tank_FreeCap

Syntax TANK_FREECAP (<*Tank ID*>)
TANK_FREECAP (TankA)

Description Returns the available capacity of the specified tank.

Use Tank_FreeCap when you need to know the available capacity of a tank.

Components

<*Tank ID*> The tank name or location index number.

Tank_DoOperation

Syntax TANK_DOOPERATION (<*Tank ID*>, <*Operation time*>)
TANK_DOOPERATION (TankA, 30)

Description Sets the state of the tank to Tank_Operation and waits for the specified operation time. ProModel sets the state to Tank_Blocked after the operation.

Use Tank_DoOperation when some activity or treatment time is necessary for the material in a tank.

Components

<*Tank ID*> The tank name or location index number.

<*Operation time*>The duration (in minutes) of the operation.

☞ **Example** After technicians add all the necessary ingredients to the mixing tank, the tank requires a 20 minute mixing time. To define this operation, enter the following statement in the subroutine for the mixing activity:

```
Tank_DoOperation(MixingTank, 20)
```

See Also *Mixing and Reactor Tanks* on page 391.

Tank_GoDown

Syntax TANK_GODOWN (<*Tank ID*>, <*Down time*>)
TANK_GODOWN (TankA, 5)

Description Sets the state of the tank to Tank_Down, waits for the specified downtime, then sets the state back to the previous setting. If you defined a downtime using the location downtime dialog, call the Tank_GoDown subroutine in the downtime logic rather than use a WAIT statement. If the downtime is for cleaning, use the Tank_DoPrep subroutine.

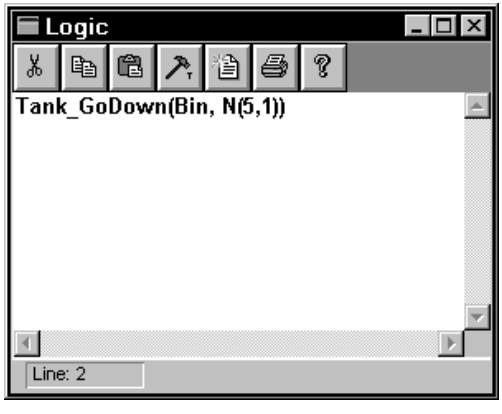
Use Tank_GoDown to shut down a tank due to equipment failure (e.g., pump failure). If the downtime occurs periodically, you can define a clock downtime in the downtime logic for the tank location and use Tank_GoDown in place of the WAIT statement.

Components

<*Tank ID*> The tank name or location index number.

<*Down time*> The duration (in minutes) of the downtime.

☞ **Example** A fill line from a dry supply bin plugs randomly according to an exponential distribution with a mean of 10 minutes. The time to unplug the line is normally distributed with a mean of 5 minutes and a standard deviation of 1 minute. To define this behavior, define a clock downtime for the bin to occur with a frequency of E(10) minutes. In the logic defined for the downtime, enter the following logic:



Tank_GoDownSched

Syntax TANK_GODOWNSCHED (<*Tank ID*>, <*Down time*>)
TANK_GODOWNSCHED (TankA, 5)

Description Sets the state of the tank to Tank_Down, waits for the specified scheduled downtime, then sets the tank state back to its *previous* setting. If you defined a scheduled downtime using the location downtime dialog, call the Tank_GoDownSched subroutine in the downtime logic rather than use a WAIT statement. If the downtime is for cleaning and you will return the tank status to *idle*, use the Tank_DoPrep subroutine.

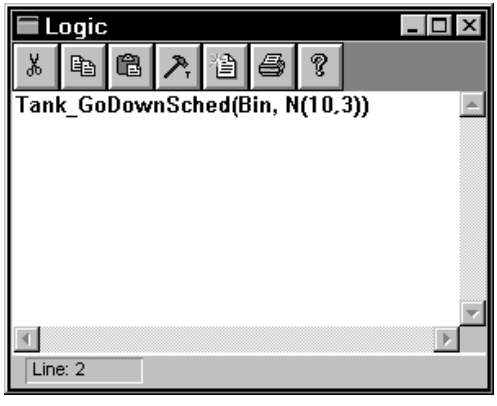
Use Tank_GoDownSched to shut down a tank for a *scheduled* task or event (e.g., interim maintenance or end of scheduled workday). Since the tank uses a *scheduled* downtime, the time lapsed during the event does not record as a downtime.

Components

<*Tank ID*> The tank name or location index number.

<*Down time*> The duration (in minutes) of the scheduled downtime.

☞ **Example** Every 4 hours, a technician must check the fill line from a dry supply bin. The time required to check the line is normally distributed with a mean of 10 minutes and a standard deviation of 3 minutes. To define this behavior, define a clock-based, scheduled downtime for the bin to occur with a frequency of 4 hours. In the logic defined for the downtime, enter the following:



Tank_DoPrep

Syntax TANK_DOPREP (<*Tank ID*>, <*Prep time*>)
TANK_DOPREP (TankA, 5)

Description Sets the state of the tank to Tank_Setup, waits for the specified time, then sets the state to Tank_Idle. Use Tank_DoPrep for cleaning activities after you empty a tank.

Use Tank_DoPrep to take a tank off line for cleaning or other preparation time.

Components

<*Tank ID*> The tank name or location index number.

<*Prep time*> The duration (in minutes) of preparation time.

☞ **Example** Workers clean a mixing tank for 30 minutes after each batch produced. To model this, enter the following logic in the mixing subroutine defined for the mixing tank:

```
Tank_Loop //causes logic to repeat continuously
  {
  [Enter mixing and transfer logic here]
  Tank_DoPrep(MixingTank, 30)
  }
```

See Also *Mixing and Reactor Tanks* on page 391.

Tank_SetState

Syntax TANK_SETSTATE (<*Tank ID*>, <*State*>)
TANK_SETSTATE (TankA, Tank_Idle)

Description Sets the state of the tank (e.g., Tank_State[<Tank ID>]) to a new state and updates the statistics since the last change of state.

Use Tank_SetState to explicitly change the state of a tank. Use Tank_SetState only if the default state changes do not adequately meet modeling needs.

Components

<*Tank ID*> The tank name or index number.

<*State*> The new state for the tank. For a list of possible tank states, see *Tank States* on page 360.

Tank_SelectOutput

Syntax TANK_SELECTOUTPUT (*<First Tank>, <Number of Tanks>, <Selection Rule>, <Maximum Level>, <Product Type>*)
TANK_SELECTOUTPUT (TankA, 3, Tank_InOrder, 5000, 0)

Description Selects an output tank from among several tanks based on a selection rule and optional product type. To use this function, list all tanks included in the selection decision together in the Location module.

Components

<First Tank> The name or location index number of the starting tank in the range.

<Number of Tanks> The number of tanks in the selection range (limit 10).

<Selection Rule> The rule for making the selection may be one of the following:

Tank_InOrder (selects the first idle tank encountered)

Tank_LongestIdle (selects the tank idle the longest)

<Maximum Level> The maximum level of the output tank before considering it for selection. Enter 0 if the output tank must be empty or idle before being considered.

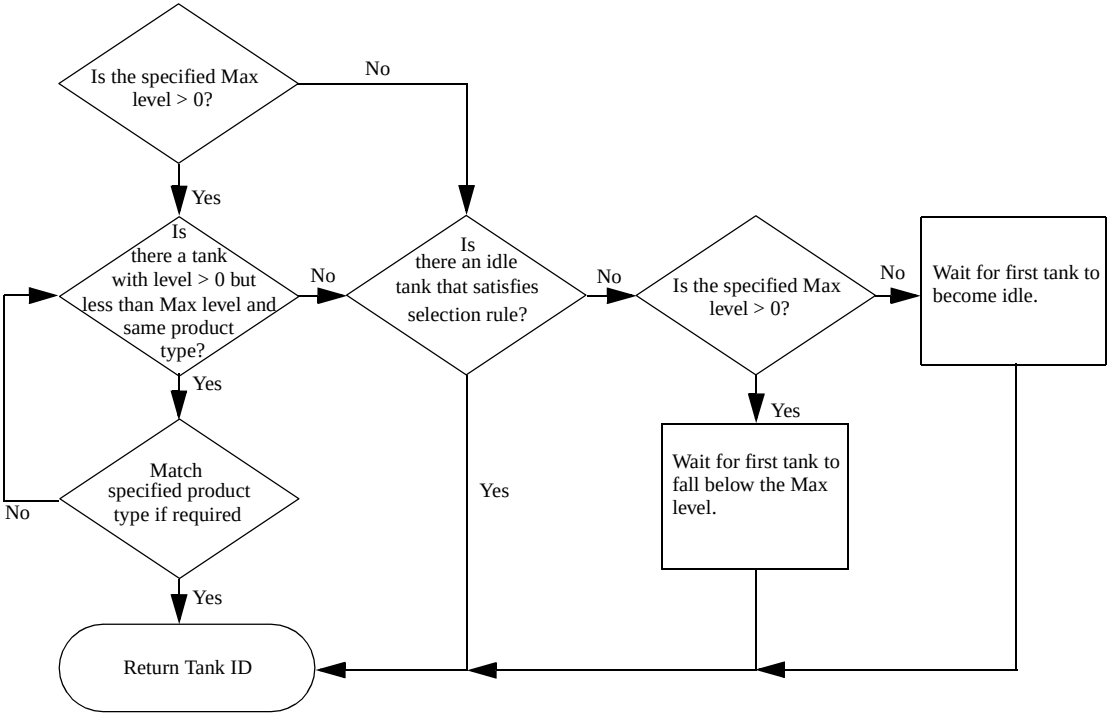
<Product Type> An integer specifying the required value of the Product array in order to select the tank. Enter 0 if the tank selection requires no product type match. (This applies only if the maximum level specified is greater than 0.)

☞ **Example** A supply tank feeds one of 3 output tanks and always gives preference first to Tank1, then to Tank2, and finally to Tank3 based on availability. Furthermore, the supply tank can select a tank only if its contents are less than 8000 gallons. To model this selection, list Tank1, Tank2, and Tank3 together (and in order) in the location module. Then define the following statement to select the tank using a local variable, Selected_Tank:

```
INT Selected_Tank
.
.
.
SelectedTank=Tank_SelectOutput(Tank1, 3, Tank_InOrder, 8000,0)
```

See diagram on following page

The diagram below shows the logic used to make a tank selection:



If you base a tank selection on product type, you must be careful to assign an appropriate integer value to the Product array element corresponding to the tank location.

See Also *Selecting from Multiple Input or Output Tanks* on page 394.

Tank_SelectInput

Syntax TANK_SELECTINPUT (*<First Tank>, <Number of Tanks>, <Selection Rule>, <Minimum Level>, <Product Type>*)
TANK_SELECTINPUT (Tank1, 5, Tank_ByOrder, 1000, 0)

Description Selects an input tank from among several tanks based on a selection rule. To use this function, you must list all tanks included in the selection together in the Location module.

Components

<First Tank> The name or location index number of the starting tank in the range.

<Number of Tanks> The number of tanks in the selection range (limit 10).

<Selection Rule> The rule for making the selection may be one of the following:

Tank_InOrder (selects the first blocked tank encountered)

Tank_LongestBlocked (selects the tank blocked the longest)

<Minimum Level> The minimum level of the input tank before considering it for selection. If the tank must be full before considering it for an input source, enter 0.

<Product Type> An integer specifying the required value for the Product array in order to select the tank. Enter 0 if the tank selection requires no product type match.

☞ **Example** A tanker arrives at a pick up station to load from one of 5 tanks depending on which tank has been full the longest. The tanker will fill from a partial tank if the tank has any contents at all (at least .1 gallons). To model the tank selection, define the following operation logic for Tanker at PickUpStation:

```
INT SelectedTank
Selected_Tank=Tank_SelectInput(Tank1, 5, Tank_LongestBlocked, .1, 0)
```

See Also *Selecting from Multiple Input or Output Tanks* on page 394.

Tank_UpdateStats

Syntax TANK_UPDATESTATS (<*Tank ID*>)
TANK_UPDATESTATS (TankA)

Description ProModel calls this subroutine automatically whenever you call any of the pre-defined subroutines that affect the tank level. If you change the value of the Tank_Level directly, call the Tank_UpdateStats subroutine afterward. This subroutine updates the current statistics on the tank and sets the state to Tank_Filling (if filling), Tank_Emptying (if emptying), Tank_Blocked (if full), or Tank_Idle (if empty).

You should not need to use this subroutine unless you defined a customized Tank_Empty, Tank_Fill, or Tank_Transfer subroutine.

Components

<*Tank ID*> The tank name or location index number.

Tank_Rate

Syntax TANK_RATE (<*FROM tank ID*>, <*TO tank ID*>)
TANK_RATE (TankA, TankB)

Description ProModel calls this subroutine automatically if you pass a 0 value as the From Rate when using the Tank_Empty or Tank_Transfer subroutine. To return the desired rate value, enter the necessary logic in the subroutine—ProModel calls the subroutine with each time step. A return value of 0 terminates the flow and returns the remaining amount in the Tank_QtyLeft variable.

Components

<*FROM Tank*> The name or location index number of the FROM tank (this value should be 0 if there is no FROM tank).

<*TO Tank*> The name or location index number of the TO tank (this value should be 0 if there is no TO tank).

☞ **Example** TankA fills with 10,000 gallons at a rate of 60 gpm until it reaches a level of 9,700 gallons. Then it fills at a rate of 30 gpm. To model this change of rate, define the following logic in the Tank_Rate subroutine:

```
IF Tank_ToID=TankA
THEN IF Tank_Level[TankA]<9700
        THEN RETURN 60
        ELSE RETURN 30
```

Now when you fill TankA, enter the following:

```
Tank_Fill(TankA, 10000, 0, 0)
```

The first 0 in the expression above causes the logic defined in the Tank_Rate subroutine to execute and determine the flow rate.

See Also *Varying the Transfer Rate* on page 396.

7.10.4 Pre-defined Data Elements

The ProModel tank submodel provides the following data elements for modeling tanks. Unless otherwise specified, all arrays are single-dimensional and of type integer. Initially, these arrays are 100 elements in size to allow for up to 100 locations. If you define more than 100 locations, you will need to enlarge the array or place tanks toward the beginning of the location list (within the first 100 locations).

Tank_Level array Stores the level of each tank. Since the values in this array directly control the tank gauge and tank statistics, the array MUST be present in every tank model.

Tank_State array Tracks the state of the tank.

Tank_Product array An optional array used to record or test the product currently at a tank.

Tank_Statistics array A two-dimensional array of type real used to record tank level statistics whenever the level changes. Generally, you will never need to reference this array since values automatically update when you use the pre-defined Tank subroutines. All times are in default time units. ProModel always gathers these statistics but reports them only if you check Basic or Time Series statistics for the tank location.

Column	Description	Reset After Warm-up
1	Last level	NC
2	Last change time	Current Time in minutes
3	Cum time-weighted level	0
4	Entries	Value of column 1
5	Max contents	Value of column 1
6	Last State Change	Current time in minutes
7	Cum time Idle	0
8	Cum time Operation	0
9	Cum time Setup	0
10	Cum time Filling	0
11	Cum time Emptying	0
12	Cum time Blocked	0
13	Cum time Down	0
14	Current downtime count	NC

As shown in the previous table, the statistics collected in the Tank_Statistics array automatically reset after any warm-up period. ProModel reports output statistics under Location statistics and Location States by Percentage. When reporting Location statistics for tanks, note the following:

- *Total Entries* The number of units (e.g., gallons, pounds) to enter the tank.
- *Avg Minutes Per Entry* Left blank since there is no individual entry for a tank.

Tank_Fills array An optional array used to track the number of transfers to a tank. This is especially useful when you activate multiple Tank_Fill or Tank_Transfer subroutines for a tank and you wish to know when the fills are complete. The user sets the value of Tank_Fills to zero before activating the subroutines, then defines a WAIT UNTIL statement after the ACTIVATE statement. The Tank_Fills array increments automatically when a Tank_Fill or a Tank_Transfer subroutine executes. See *Mixing and Reactor Tanks* on page 391 for additional information.

Statistics

Note

TS = Tank_Statistics array
n = Location index number of tank

Calculating Location Statistics for Tanks

Entries = TS [n, 4]
Avg. Time per Entry = (not applicable)
Avg. Contents = TS[n,3] / Scheduled Time
Max Contents = TS [n,5]
Current Contents = TS [n,1]
Utilization=100 x TS[n,3] / (Capacity x Sim Time)

Calculating Location State Statistics for Tanks

%Operation = 100 x TS[n, 8] / Scheduled Time
%Setup = 100 x TS[n, 9] / Scheduled Time
%Idle = 100 x TS[n, 7] / Scheduled Time
%Waiting = 100 - Sum of other percentages
%Blocked = 100 x TS[n, 12] / Scheduled Time
%Down = 100 x TS[n, 13] / Scheduled Time

To gather statistics on how much of a particular product was processed, you may define variables to record product processing during the simulation.

Sample statistics

```
--------------------------------------------------------------------------------
General Report
Output from N:\NR\TANKS\Tankfil1.mod
Date: Sept/20/1998   Time: 11:47:36 AM
--------------------------------------------------------------------------------
Scenario        : Normal Run
Replication     : 1 of 1
Simulation Time : 40 hr
--------------------------------------------------------------------------------
LOCATIONS
                                          Average
Location  Scheduled             Total     Minutes   Average   Maximum   Current
Name          Hours  Capacity  Entries  Per Entry  Contents  Contents  Contents  % Util
--------  ---------  --------  -------  ---------  --------  --------  --------  ------
Tank1            40      1000        1          -         0         0         0    0.00
Tank2            40      1000        0          -         0         0         0    0.00
Tank3            40      1000        0          -         0         0         0    0.00
Pipe             40         1        0          -         0         0         0    0.00
Delivery         40         1        0   0.000000         0         0         0    0.00

LOCATION STATES BY PERCENTAGE (Single Capacity/Tanks)
Location  Scheduled          %      %       %        %        %     %
Name          Hours  Operation  Setup    Idle  Waiting  Blocked  Down
--------  ---------  ---------  -----  ------  -------  -------  ----
Pipe             40       0.00   0.00  100.00     0.00     0.00  0.00
Delivery         40       0.00   0.00  100.00     0.00     0.00  0.00
Tank1            40       0.00   0.00   16.67    66.67    16.67  0.00
Tank2            40       0.00   0.00   19.58    60.42    20.00  0.00
Tank3            40       0.00   0.00   24.58    75.42     0.00  0.00

FAILED ARRIVALS
Entity  Location   Total
Name    Name      Failed
------  --------  ------
Tanker  Delivery       0

ENTITY ACTIVITY
                          Average  Average    Average    Average  Average
                  Current Minutes  Minutes    Minutes    Minutes  Minutes
Entity  Total   Quantity       In  In Move   Wait For         In
Name    Exits  In System   System    Logic  Res, etc.  Operation  Blocked
------  -----  ---------  -------  -------  ---------  ---------  -------
Tanker      0          0        -        -          -          -        -

ENTITY STATES BY PERCENTAGE
              %          %
Entity  In Move   Wait For             %        %
Name      Logic  Res, etc.  In Operation  Blocked
------  -------  ---------  ------------  -------
```

7.10.5 Defining Tank Control Subroutines

Unlike defining entity activity at a location (defined in the Processing module), modeling tank location activity requires the use of subroutines. Many of these subroutines are user-defined and called using the ACTIVATE statement. Though you generally activate them from the initialization logic, you may also activate them from another tank subroutine. Tank subroutines consist of logic defined to control when, where, and how much to empty, fill, or transfer from a tank. Often, these subroutines require the use of WAIT UNTIL statements to monitor conditions (e.g., the tank level or state) before making a transfer and may include delays for mixing or cleaning.

At a minimum, you should define a separate control subroutine for any logic that executes independently of any other logic. For example, if TankA fills TankB at the same time TankB transfers to some other tank, you should define two separate subroutines since both sets of logic must be capable of executing independently of each other. On the other hand, if the logic associated with two tanks is interdependent, only one control subroutine is necessary. For example, if TankA fills TankB while TankB waits, then TankB pumps out while tank A waits, you need only a single control subroutine since you control both tanks by a single logic sequence. If a single tank feeds several other tanks independently, you would need a separate subroutine to control each output. In most cases, you will need at least one control subroutine per tank and, in certain situations, you may wish to use a hierarchical control system (i.e., a master or supervisory control subroutine) to activate subordinate subroutines.

Most tank control subroutines should be activated subroutines. In contrast to *called* subroutines, *activated* subroutines use the ACTIVATE statement and cause the logic activating the subroutine to continue independently of the activated subroutine. This allows you to execute multiple control subroutines concurrently. Multiple tanks with identical control logic may share the same control subroutine if you activate the subroutine for each tank and pass the tank ID as a parameter.

One of the keys to modeling interactive tank behavior is to effectively use WAIT UNTIL statements. When you use WAIT UNTIL statements based on the Tank_Level array, use them *sparingly* since this array changes frequently and may slow the simulation.

7.10.6 Examples of Tank Control Logic

The following examples show how to model different tank and flow situations. For full models illustrating these situations, see PROMOD4\MODELS\REFS.

Filling from an Entity

A typical tank modeling situation is the arrival of an entity (e.g., a tanker or other vehicle) to deliver its contents to a tank. To model this situation, define an arrival or routing for the entity, causing it to enter the location where it will make its delivery. In the entity processing logic at the delivery location, call the Tank_Fill subroutine. By *calling* rather than *activating* the subroutine, you will detain the delivering entity until Tank_Fill executes. Note that the material does NOT route from the delivery location to the tank. Instead, the Tank_Fill subroutine simply fills the tank with a specified quantity while the entity waits. Unless the quantity is a constant amount, it is usually a good idea to use an entity attribute to store this quantity value. After filling the contents into the tank, the entity is free to continue processing.

To illustrate how an entity might transfer its contents to a tank, suppose an entity, Tanker, arrives at a location, Delivery, carrying a quantity of gallons stored in an entity attribute called Tanker_Qty. The tanker discharges its contents into a tank, ReceivingTank, at a rate of 200 gallons per minute. Once the ReceivingTank becomes full, the level must drop to 1000 gallons before filling resumes. Since the entity is tied up while it discharges into the tank, use the following statement in the processing logic for Tanker at Delivery to define the logic used to fill the tank:

```
Tank_Fill (ReceivingTank, Tanker_Qty, 200, 1000)
```

The above statement causes each arriving tanker to wait until the quantity stored in its Tanker_Qty attribute adds to the ReceivingTank. Once the tanker delivers this quantity, it is free to execute the routing defined for it at the Delivery location.

Initializing and Replenishing Supply Tanks

A supply tank is an originating tank that is a source of raw material for one or more downstream tanks. Often, supply tanks contain ingredients that feed into a mixing tank or hold chemicals that feed into a reactor. Typically, you replenish a supply tank when it gets low and make it available for use whenever it has an adequate supply. If you always stock the supply tank and it is always available for use, you do NOT need to model it since it poses no constraint on the process. You may set supply tanks to an initial level at the start of the simulation in the initialization logic, then use them as needed by a mixing or other downstream tank. To initialize the level in a supply tank, enter the following statement in the initialization logic:

```
Tank_SetLevel (<supply tank>, <qty>)
```

If, for example, you wanted to begin the simulation with the supply tank, WaterTank, filled with 800 gallons of water, you would enter:

```
Tank_SetLevel (WaterTank, 800)
```

To gradually fill or refill a supply tank whenever it drops below a trigger level, use the Tank_Fill subroutine with a large fill quantity and an appropriate resume level. For example, the following statement will continue pumping up to 999999 units into TankA at a rate of 200 units per minute. Whenever the tank becomes full, it must drop to 400 units before filling resumes.

```
Tank_Fill (TankA, 999999, 200, 400)
```

Mixing and Reactor Tanks

Mixing and reactor tanks receive material usually from one or more supply tanks. Once it receives the material, the tank may require a mixing or other reaction time. To illustrate, suppose we have two tanks (Tank1 and Tank2) supplying ingredients to a tank called MixingTank. First, workers pump 2000 gallons of a liquid from Tank1 at 50 gallons per minute followed by the transfer of 300 pounds of dry mix from Tank2 at 20 pounds a minute (the dry mix adds .2 gallons to the level of the MixingTank for every pound transferred, equating to 4 gallons per minute). The ingredients then mix for 15 minutes before transferring to an idle storage tank. After transferring the mix, workers must clean the MixingTank for 50 minutes to prepare it for the next mixing cycle.

The control logic for the mixing tank should be a subroutine activated from the initialization logic which continues to loop throughout the simulation. The subroutine logic might appear as follows:

```
Tank_Loop // Continues to loop until the simulation ends
    BEGIN
    Tank_Transfer (Tank1, MixingTank, 2000, 50, 0, 0)
    Tank_Transfer (Tank2, MixingTank, 300, 20, 4, 0)
    Tank_DoOperation (MixingTank, 15) // Mixing time
    Wait Until Tank_State [StorageTank] = Tank_Idle /* Waits for
        storage tank availability*/
Tank_Transfer (MixingTank, StorageTank, Tank_Level[MixingTank],
        40, 0, 0)
    Tank_Prep (MixingTank, 50) // Clean mixing tank for 50 minutes.
    END
```

If the ingredients feed into the mixing tank at the same time rather than sequentially, activate the Tank_Transfer subroutines for the mixing tank and monitor the Tank_Fills array to know which ingredients enter into the tank. For simultaneous fills, replace the first two transfer statements following the BEGIN statement in the previous subroutine with the following logic:

```
Tank_Fills[MixingTank]=0
ACTIVATE Tank_Transfer(Tank1, MixingTank, 2000, 50, 0, 0)
ACTIVATE Tank_Transfer(Tank2, MixingTank, 300, 20, 4, 0)
WAIT UNTIL Tank_Fills[MixingTank]=2
.
.
.
```

Emptying to an Entity

Often, tanks deliver material to discrete entities such as containers (or perhaps the material itself converts to discrete entities through a solidification or consolidation process). In either case, you can draw from the delivery tank using the Tank_Empty subroutine if outflow is gradual and defined by a flow rate, or the Tank_Dec subroutine if the output occurs in discrete intervals based on a bottling or packaging time.

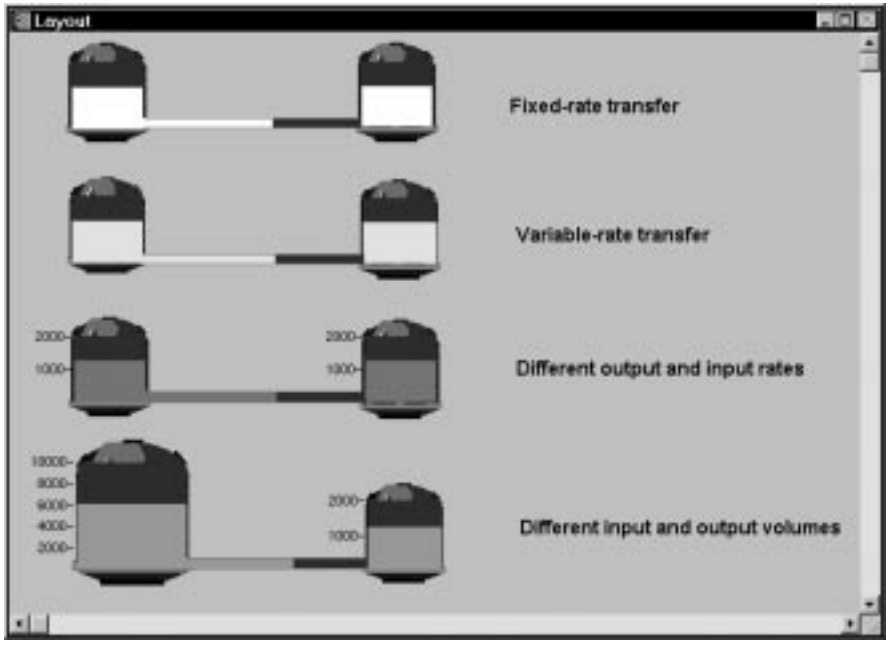
To output material from a tank without modeling the entity to which it outputs, call the Tank_Empty or Tank_Dec subroutine. To transfer material from a tank to entities arriving at a filling station (remember, the filling station itself is NOT a tank), route the entities to the filling station using a SEND or other routing rule, then call the Tank_Empty or Tank_Dec subroutine.

If using the Tank_Dec subroutine, the entity should wait for the fill time *before* decreasing the tank level since Tank_Dec happens *instantly.* For example, if a bottling operation fills a 2 gallon container every 6 seconds, define the following processing logic for the container at the fill station:

```
Wait 6 sec
Tank_Dec (Filler, 2)
```

If the delivery tank has insufficient contents to decrease the level by the specified amount, the processing will automatically pause until enough material is available. Once the specified quantity empties, the entity can continue processing. To create an entity as the result of an emptying operation, define an activated subroutine that empties the desired quantity, then execute an ORDER statement. This will create a new entity at the filling station.

Tank Transfers



When transferring from one tank to another, you must determine whether the source tank makes the decision to transfer to the destination tank (a push approach) or whether the destination tank makes the decision to draw material from a source tank (a pull approach). You should define a control subroutine from the perspective of the tank that makes the decision. If the model requires no tank selection, specify a WAIT UNTIL statement to wait until the FROM or TO tank satisfies the condition required for transfer. For example, if a source tank makes the decision to transfer to a destination tank whenever the destination tank becomes idle, enter the following statement in the subroutine:

```
Wait Until Tank_State [<destination tank ID>] = Tank_Idle
```

If the destination tank makes the decision to transfer (a pull approach), you should base the WAIT UNTIL statement on a required condition for the source tank as follows:

```
Wait Until Tank_State [<source tank ID>] = Tank_Blocked
```

Following the WAIT UNTIL statement, call the Tank_Transfer, Tank_TransferUpTo, or Tank_TransferDownTo subroutine to transfer from the source tank to the destination tank.

To illustrate how to define a tank transfer using a pull approach, suppose that TankB requires 1000 gallons from TankA whenever TankB becomes empty. TankB will draw material from TankA only when TankA has a minimum level of 1000 gallons. The subroutine to define this logic might appear as follows:

```
Tank_Loop   // Causes the logic to be repeated during entire simulation.
    Begin
    Wait Until Tank_Level[TankA] >= 1000 /*Wait for TankA to reach
        1000 gallons*/
    Tank_Transfer (TankA, TankB, 1000, 200,0,0) /* Transfer 1000 gal
        to TankB at 200 gpm*/

    [Enter TankB processing and emptying logic here]

    End
```

Note To select from among multiple input or output tanks, activate this subroutine in the initialization logic.

Selecting from Multiple Input or Output Tanks

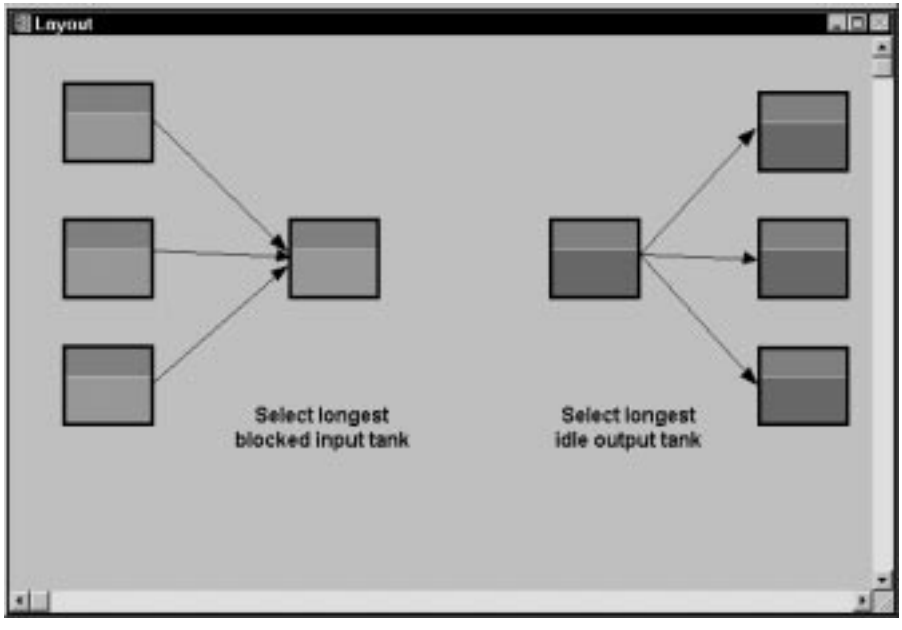
To enable one or more tanks to select from several input or output tanks, use the predefined subroutine Tank_SelectInput or Tank_SelectOutput (see subroutine descriptions). ProModel bases tank selection on which tank is ready to transfer or receive and that has the same ProductType array value. You must list the tank selections together in the Location module.

For example, if TankX selects from among three input tanks (Tank1, Tank2, and Tank3) based on which input tank has waited the longest to discharge its contents, you would enter the following logic in the control subroutine defined for TankX:

```
Int SelectedTank
SelectedTank = Tank_SelectInput(TankX, Tank1, 3,
    Tank_LongestBlocked, 0)
```

The first statement defines a local variable, SelectedTank, used to assign which tank you select. The second statement calls the SelectInput subroutine specifying that TankX is to select one of three tanks beginning with Tank1. Tank_LongestBlocked causes TankX to select the tank blocked the longest (i.e., tank is full or waiting). Entering 0 at the end prevents selecting a full tank. If no tank is full, the statement does not execute until one of the input tanks fills. With a tank ID assigned to SelectedTank, you can call a transfer subroutine to make the transfer.

For output tanks, you would define similar logic but include Tank_Select*Output* instead of Tank_Select*Input*.



Split Transfers

Sometimes it is necessary to use a tank or separator to split the flow to several output tanks. To define the concurrent transfer of material from one tank to multiple tanks, define an activated subroutine for each transfer. Suppose, for example, that when TankA fills it begins transferring to TankB at a rate of 30 gpm and to TankC at a rate of 40 gpm. To know when both transfers are complete, define a global variable (e.g., TransferDone) which increments at the end of each transfer. Defining the following logic would initiate this split transfer once TankA is full:

```
ACTIVATE TransferToB () // initiates transfers from A to B
ACTIVATE TransferToC () // initiates transfers from A to C
WAIT UNTIL TransferDone = 2 // Wait until transfers are complete
TransferDone = 0 // reset for next transfer
```

The subroutines TransferToB and TransferToC would each execute a Tank_TransferDownTo command followed by a statement incrementing the value of TransferDone. For example, the logic for TransferToB would be as follows:

```
TransferDownTo(TankA, TankB, 0, 30, 0)
INC TransferDone
```

Varying the Transfer Rate

The transfer or empty rate can change dynamically during an empty, fill, or transfer. To vary the rate of flow, pass 0 as the flow rate when calling any of the transfer, fill, or empty subroutines. This calls the Tank_Rate subroutine automatically with each time step. You should modify the Tank_Rate subroutine so that it returns the appropriate rate value.

Suppose, for example, that TankA transfers to TankB at a rate that decreases from 150 gpm to 50 gpm when the level of TankB reaches 4000. To achieve this, pass 0 as the From Rate when you call the transfer subroutine, then enter the following logic in the Tank_Rate subroutine:

```
IF (Tank_FromID = TankA) AND (Tank_ToID = TankB)
THEN IF Tank_Level[TankB] >= 4000
THEN RETURN 50
ELSE RETURN 150
```

Dynamically Suspending Flow

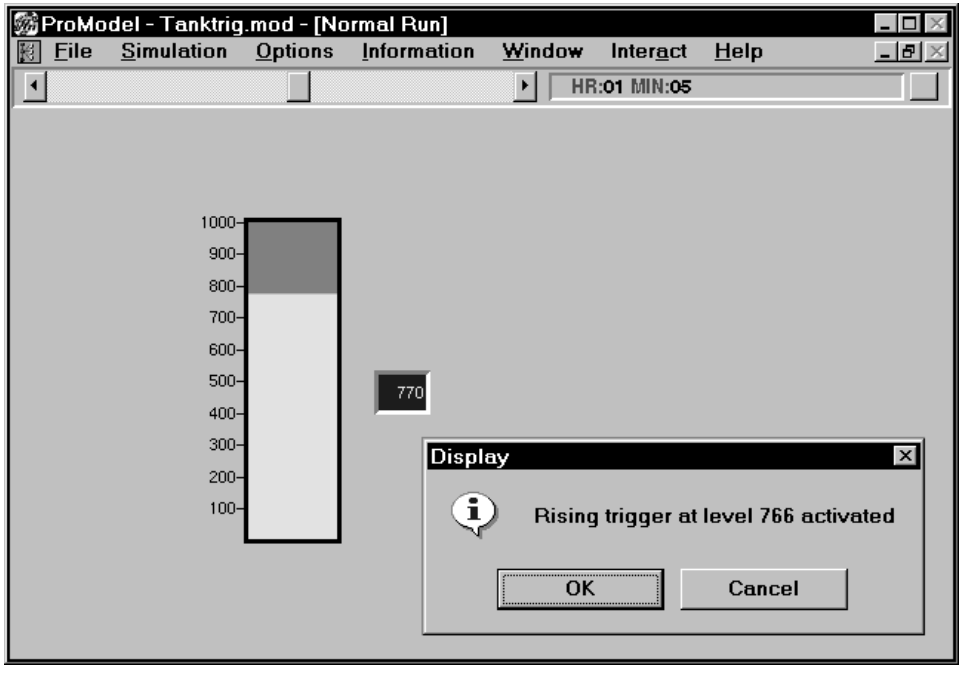
To momentarily interrupt flow into or out of a tank, use the Tank_GoDown subroutine or set the state of the tank to down (Tank_SetState = Tank_Down). This typically happens if a pump fails but may occur in other situations.

Dynamically Terminating a Flow

Normally, the flow into or out of a tank stops once you reach the desired quantity or level. However, in some situations you may wish to terminate a transfer if some event or condition occurs that you cannot predetermine (e.g., a decision to divert flow to a preferred outlet tank that just became available). In this case, you can turn off the flow into or out of a tank by specifying a variable transfer rate instead of a fixed transfer rate (see *Varying the Transfer Rate*). A variable transfer uses the Tank_Rate subroutine to determine the rate for each time step—to terminate a transfer, return a rate value of 0.

Defining Trigger Levels

A trigger level is a level to which material in a tank either falls or rises and triggers some action. To continuously monitor when a tank reaches a trigger level, define and activate a trigger subroutine in the initialization logic. The subroutine should call Tank_RiseTrigger or Tank_FallTrigger depending on whether the associated action should execute when the tank level *rises* or *falls* to a certain level.



To show how to define a trigger subroutine, suppose that whenever TankA rises to 2000 gallons, an entity called Truck travels to a location called Pickup. The logic for this trigger subroutine might look as follows:

```
Tank_Loop // causes logic to repeat for entire simulation.
    Begin
    Tank_RiseTrigger (TankA, 2000) /* waits for TankA to rise to 2000
        units*/
    Order 1 Truck to Pickup // order a Truck to Pickup
    End
```

Once the tank reaches the trigger level, the Tank_RiseTrigger subroutine prevents further triggering until the level drops back below the trigger level first.

When you use trigger subroutines, use them *sparingly* because they are CPU *intensive*. Every time the tank level changes, ProModel tests to see if the tank reached the trigger level. Trigger subroutines are often unnecessary because, unlike an actual tank

where sensors report the tank level, you directly control how much to pump into a tank. For instance, an alternative way to model the previous example without using a triggering subroutine would be to call the Tank_TransferToLevel subroutine to first fill the tank to the 2000 unit level, order the Truck entity and then transfer the rest.

Processing Multiple Products

Where you must track several different products through one or more tanks, it may be useful to define macros for naming each product type. For example, setting ProductA equal to 1 and ProductB equal to 2 will improve the readability of the model. To track which product a particular tank is processing, ProModel uses a pre-defined integer array called Tank_Product—the user is responsible for maintaining the array values. If, for example, ProductA begins pumping into Tank1, enter the following after you assign an integer value to ProductA in the Macros module to distinguish it from other products:

```
Tank_Product [Tank1] = ProductA
```

Showing Pipes

To show pipes connecting the tanks, use paths or background graphics. If you desire to show the material in the pipe, use a long, skinny tank with a capacity of 1 to represent the pipe. You can set the level of this tank to 0 or 1 to show product flow. For example, suppose we define a tank location called Pipe used to represent the connection between Tank1 and Tank2. Whenever transferring from Tank1 to Tank2, you would enter the following:

```
Tank_SetLevel (Pipe,1)
Tank_Transfer (Tank1, Tank2, ….)
Tank_SetLevel (Pipe, 0)
```

High-Rate Entity Processing

For systems that process entities at rates higher than one hundred units per minute, using discrete entities could make the simulation extremely slow. For this reason, ProModel uses tanks. To use a tank to model high-rate processing, think of the tank as a buffer where the tank level represents the number of items in the buffer. For example, suppose that bottles feed through a filling station at a rate of 110 per minute. The input buffer, FillerInput, has a capacity of 1200 bottles and the output buffer, FillerOutput, has a capacity of 2000 bottles. If FillerOutput is full, processing stops until the quantity in the output buffer drops to 1500 bottles. An arriving container feeds quantities of 200 bottles to the FillerInput location and it takes 1 minute to unload the container. When the filling station fills 50 bottles, workers put the bottles into a box (represented by an entity) and ship them. Since workers load the boxes as soon as the bottles complete the filling process, there is no delay time involved.

The operation logic for the container at the arriving location would be as follows:

```
WAIT 1 min
Tank_Inc (FillerInput, 200)
```

To model the processing of bottles from FillerInput to FillerOutput, enter the following statement in the model initialization logic.

```
ACTIVATE Tank_Transfer(FillerInput, FillerOutput, 999999, 110, 0, 1500)
```

This statement causes the FillerInput tank to transfer bottles to FillerOutput at a rate of 110 per minute whenever there are bottles in FillerInput and capacity available in FillerOutput. The resume level is 1500. (Up to 999999 bottles will transfer.)

To model the creation of a 50-bottle box each time the filling station fills 50 bottles, define and activate the following subroutine in the model initialization logic:

```
Tank Loop //causes logic to repeat continuously
    {
    Tank_Dec(FillerOutput, 50)
    Order 1 Box to Shipping
    }
```

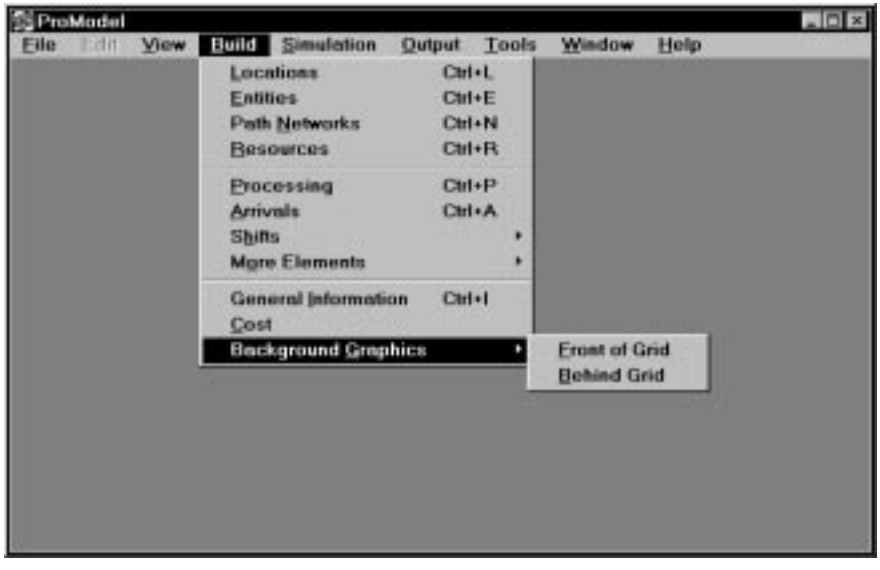
Note The Tank_Dec statement automatically removes 50 bottles from FillerOutput whenever there are at least 50 bottles available.

Special Notes

- Since tank models do not stop automatically when there are no more entities or scheduled arrivals, remember to define a run length or a STOP statement.
- When you activate a subroutine, it doesn't process until the current logic (the one activating the subroutine) finishes or becomes blocked. It you want the activated subroutine to process *first*, enter “WAIT 0” after the ACTIVATE statement.
- Do not define a local variable inside of a Tank_Loop since the loop will create the variable multiple times.
- Make sure all IF...THEN logic and WAIT UNTIL statements based on the Tank_Level array use the “>=” or “<=” operator and not just an “=” operator. (This is because flow occurs in increments and you can’t check for an exact value.)
- Tanks are not legal in multi-unit locations or in locations containing a conveyor or queue.

7.11 Background Graphics

Background graphics allow you to enhance a model by adding a background to the animation. A background could show a floor-plan of a factory or any item that is not part of a location, entity, or resource. Backgrounds can be created using the tools in the Background Graphics Editor or by importing an existing background from another application, such as AutoCAD. Imported graphics must have been saved in one of the following formats BMP, WMF, GIF, or PCX. The Background Graphics Editor is accessed from the Build menu as shown below.



There are two modes for editing in the Background Graphics Editor: Front of Grid and Behind Grid. Generally, most graphics should be laid out in front of the grid. Graphics placed behind the grid should be reserved for large objects such as walls or imported backgrounds.

How To

Create or edit background graphics:

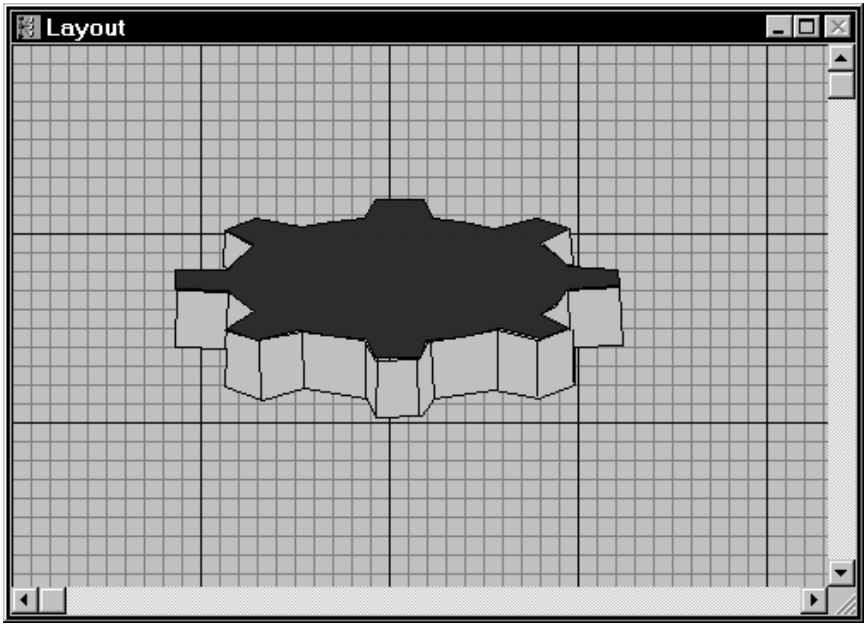
1. Select **Background Graphics** from the **Build** menu.
2. Select **Front of Grid** or **Behind Grid** depending on the mode desired.

7.11.1 Background Graphics Editor Modes

ProModel gives you the option of placing the background graphic in front of or behind the grid. This is useful when you want to view the imported background graphic, but you also want to see the grid for drawing and sizing objects on the imported background graphic.

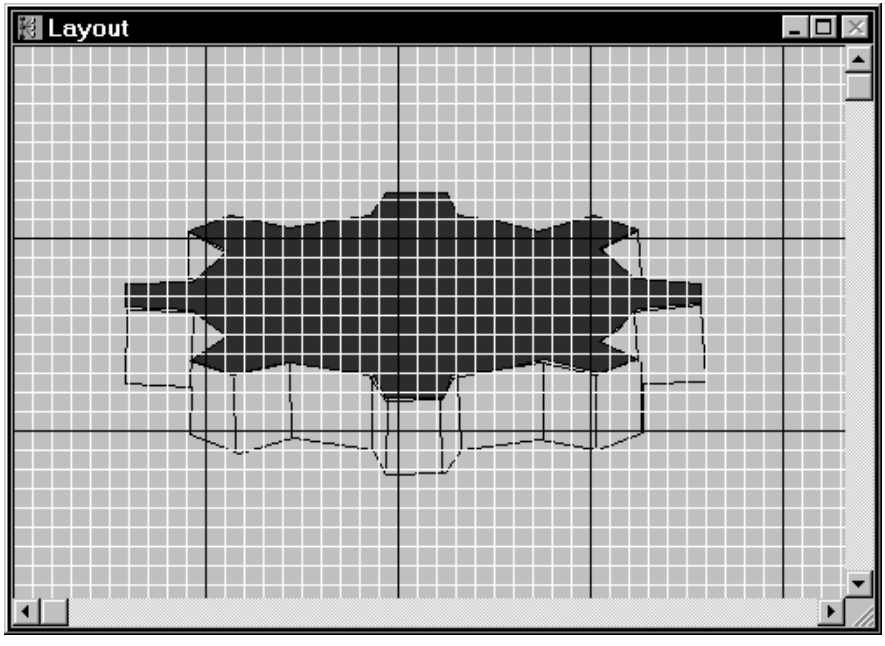
Front of Grid Mode

Creating graphics in this mode places the background graphic in front of the grid as shown below.



Behind Grid Mode

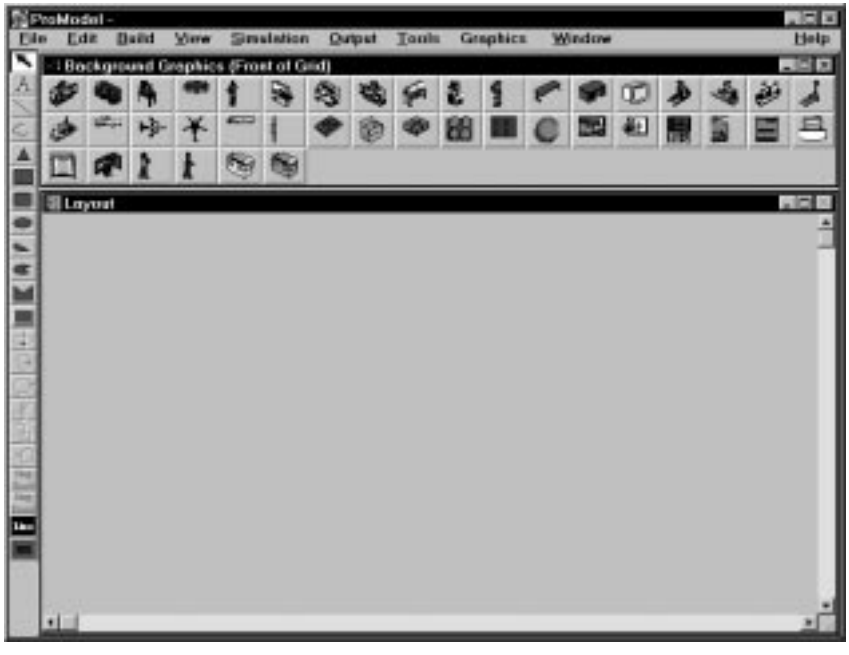
Creating graphics in this mode places the background graphic behind the grid as shown below.



All floor-plans should be imported, designed or drawn in the “Behind Grid Mode” and all remaining graphics placed on the layout in “Front of Grid Mode.” This prevents you from accidentally moving the background graphic while editing other elements of the model.

7.11.2 Background Graphics Editor

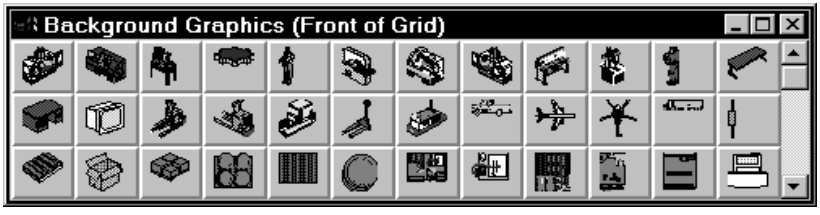
The Background Graphics Editor allows you to place icons, text and other graphic shapes on the layout behind locations and other system element graphics. The arrangement of the two windows and button bar is shown below.



- The Library Graphics window, containing all the icons in the current graphic library, is located at the top of the workspace.
- The Tools button bar, where you may select a tool for creating and editing graphic shapes, is located at the left of the workspace.
- The Layout window, where all creating and editing of graphic shapes is done, is located in the lower right portion of the workspace.

Library Graphics Window

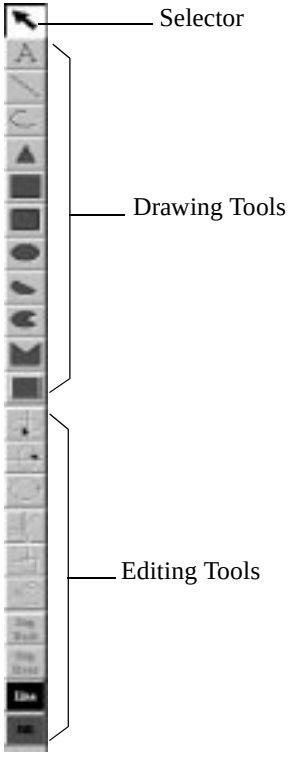
The Library Graphics window contains the icons of the current graphic library file, specified in the General Information dialog. These icons may be placed on the Layout in the same way as other objects.



You may size the window as desired, or use the scroll bar shown above to scroll through the available icons.

Tools Button Bar

The Tools button bar contains the tools necessary to create objects various shapes. It also contains tools for editing those objects including flip, rotate, and cut.



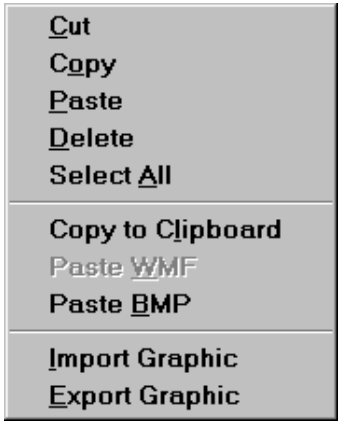
The Background Graphics Editor Tools button bar is the nearly the same Tools button bar used in the Graphic Editor. The only difference is that the Background Graphics Editor Tools button bar does not contain an entity spot or status light tool. For more information see *Graphic Tools Button Bar* on page 535.

Edit Menu

Use the Edit menu for selecting and duplicating the graphic objects in the current Background Graphics mode. You may also use it to exchange graphics with other applications. To use the Edit menu functions, select the object you wish to edit by clicking on it in the Layout window.

The first four functions apply to the currently selected object. To select multiple objects, hold the shift key while selecting an object. Alternatively you can drag a rect-

angle encompassing the objects you want selected. To deselect one of several selected objects, click on the selected object while holding the shift key.



Cut Removes the selected object(s) and makes a temporary copy that may be pasted back into the edit window later.

Copy Makes a temporary copy of the selected object(s) for later pasting.

Paste Adds the most recently cut or copied object(s) to the Layout window.

Delete Deletes the selected background graphic from the Layout window.

Select All Selects all of the graphic objects in the current mode.

Copy to Clipboard Copies all graphic objects in the current mode to the clipboard as a bitmap or windows metafile so they can be pasted into another application such as a word processor.

Paste WMF Pastes a Windows metafile (WMF) from the Windows clipboard into the Layout window. You must have previously copied a Windows metafile to the Windows clipboard.

Paste BMP Pastes a bitmap file (BMP) from the Windows clipboard into the Layout window. You must have previously copied a bitmap file to the Windows clipboard.

Import Graphic Imports a WMF, BMP, PCX, or GIF file into the Layout window.

Export Graphic Exports all graphic objects in the current mode to a WMF or BMP file.

Importing a Graphic

Importing a background graphic can bring reality into the model. For example, if a layout is created in a graphic package, it may be desirable to import the entire layout rather than create it in ProModel. This is done by saving the file in a graphic package, such as AutoCAD, as a WMF, BMP, PCX, or GIF file and importing the graphic into ProModel. This can save you an extensive amount of time.

How To — Import a background graphic into the layout:

1. In a graphics application, save the graphic in one of the following formats, WMF, BMP, PCX, or GIF.
2. Select **Background Graphics** from the **Build** menu.
3. Select **Front of Grid** or **Behind Grid**.
4. Select **Import Graphic** from the **Edit** Menu.
5. Enter the name of the graphic you would like to import.
6. Select **OK** to close the import graphic dialog box. The graphic will then appear in the layout window. The upper left corner of the imported graphic will align with the upper left corner of the layout window.

How To — To move an imported background graphic:

- Place the cursor on the imported background graphic and drag it to the desired location in the layout.

How To — To size an imported background graphic:

1. Place the cursor on one of the four corners of the imported background graphic. The cursor becomes a cross-hair at this point.
2. Drag the cursor to size the background graphic as desired.

Note Once imported, the background graphic is not a separate file from the model. It is included in the model. Therefore, when moving or copying a model file from one directory to another, it is not necessary to move or copy the imported background graphic file as well. On the other hand, if the external graphic file is changed, it must be re-imported to update the model layout.

Exporting a Graphic

In some cases, it is desirable to export a graphic created in ProModel for use in another application. ProModel will export all objects in the current mode (In Front of Grid or Behind Grid) as one graphic to a WMF or BMP file.

How To **Export a graphic:**

1. Select **Export Graphic** from the **Edit** menu.
2. Enter a valid DOS name for the graphic in the resulting dialog box, such as forklift.bmp.
3. Click the OK button in the Export Graphic dialog box.

Graphics Menu

The Graphics Menu allows you to flip and rotate the selected graphic object(s) in the layout window. It also allows you to specify whether you want the graphic to be behind or in front of the grid. Additionally, it allows you to group selected graphic objects together into a single graphic. Finally, it provides the option to define line styles, fill patterns, line color, and fill color.

Flip Horizontal Horizontally flips the selected object(s).

Flip Vertical Vertically flips the selected object(s).

Rotate Rotates the selected object(s) 90 degrees clockwise. This does not apply for *non*-true-type fonts.

Behind Grid Moves the selected object in the layout window behind the grid. Once this is done, you must go to Behind Grid mode to edit the graphic.

Front of Grid Moves the selected object in the layout window in front of the grid. Once this is done, you must go to Front of Grid mode to edit the graphic.

Group Combines or groups several graphic objects into a single graphic so they may be sized and edited together.

Ungroup Ungroups several grouped graphic objects so they may be edited individually.

Line Styles Allows the user to define the line style including transparent, dashed, line thickness, and optional arrowheads on either end of the line. If any objects are selected, the line styles of the selected objects are changed.

Fill Patterns Allows the user to define the fill pattern for solid objects including slant, grid, crosshatch, backward slant, horizontal, vertical, transparent, solid, vertical gradient, and horizontal gradient. If any objects are selected, the fill patterns of the selected objects are changed.

Line Color Allows the user to define the line color and create custom colors. If any objects are selected, the line color of the selected objects are changed.

Fill Color Allows the user to define the fill color and create custom colors for solid objects. If any objects are selected, the fill color of the selected objects are changed.

Note All functions in the Graphic menu of Background Graphics are nearly the same functions described in the Graphic Editor Graphic menu. Differences are noted below for moving a graphic behind the grid and in front of the grid. See *Graphic Editor on page 515* for more information on the above functions.

How To

Move a graphic behind the grid:

1. Select the graphic on the layout using the selector.
2. Select **Behind Grid** from the **Graphics** menu.

How To

Move a graphic in front of the grid:

1. Select the graphic on the layout using the selector.
2. Select **Front of Grid** from the **Graphics** menu.