user guide

Disclaimer

The information in this guide is provided by ProModel Corporation to document ProModel. The contents of this manual are subject to change without notice and does not represent a commitment on the part of ProModel Corporation. The software described in this guide is applied under a license agreement and may be copied only under the terms of the license agreement. No part of this guide may be reproduced, transmitted, or distributed by any means, electronic or mechanical, for purposes other than the owner's personal use without express written permission from ProModel Corporation.

Copyright Information

Printed in the United States of America

1/06

Table of Contents

##

##

Chapter 10: Reports and Graphs 373

Chapter 11: Language Elements and Expressions 403

Chapter 14: Statements and Functions 439

Introduction

This chapter will introduce you to the user guide, and provide information regarding assumptions this guide makes. You may also find information on support and consulting services in this chapter.

About the User Guide

The ProModel User Guide is designed as a reference to guide you through the process of building, running, and viewing the results of simulation models. The guide contains detailed information on the use of features and capabilities found within ProModel and serves as a compliment to the product training.

Chapter 1 Getting Started

Welcomes you to ProModel and provides information on getting started with the model building process.

Chapter 2 Installation and Registration

Provides step-by-step instructions on how to install ProModel.

Chapter 3 Planning the Model

Introduces you to the theory behind simulation modeling and gives an overview of model building procedure. This chapter outlines six steps to help you plan and create a successful, valid model. It then provides an example scenario, which designs a model from start to finish.

Chapter 4 Modeling Environment

Familiarizes your with the ProModel modeling environment. You will learn how to use many of the menus and windows found in ProModel.

Chapter 5 Building the Model: general elements

Brings you face to face with basic modeling components. Discussion revolves around the nuances of creating and working with locations, entities, path networks, resources, processing records, arrivals, shifts, costs, background graphics, and more.

Chapter 6 Building the Model: advanced elements

Advanced elements ensure that the models you create reflect the exact behavior of your system. Contributing elements include attributes, variables, arrays, spreadsheets, macros, run-time interfaces, subroutines, arrival cycles, table functions, user-defined distributions, external files, and streams.

Chapter 7
Building the Logic

Applying intelligence and decision-making capabilities to your models makes them true to life. The logic builder helps you create and implement logic for operations, routings, shifts, breaks, and more.

Chapter 8
Using Auxiliary Tools

Auxiliary tools allow you to work with many aspects of your model. You may create and modify graphics, search for expressions, or examine which statistical distributions best suit your modeling needs.

Chapter 9
Running the Model

Running the model is the most exciting part of the modeling process. This chapter discusses the various conditions under which you may run a model and how to ensure that the model contains no logical errors.

Chapter 10
Reports and Graphs

With the simulation complete, you are ready to examine the results from the model. Options available include reports, graphs, spreadsheets, and customized output templates.

Chapter 11
Language Elements and Expressions

Describes the basics of the language used to define model objects and logic, as well as expressions and the role they play in creating models. Discussion focuses on names, reserved words, numbers, strings, operators, and basic syntactical structure, as well as operator precedence and numeric, Boolean, time, and string expressions.

Chapter 12
Routing Rules

Provides syntax and examples for the rules you will use to route between locations in the model.

Chapter 13
Logic Elements

Introduces the different types of functions and statements available to use in building logic for your model and identifies where you may use them.

Chapter 14
Statements and Functions

Provides syntax and examples for the statements and functions you will use to apply intelligence and decision-making capabilities to your model.

Appendix A

Presents a categorized list of all expressions, statements, and functions with a table identifying which fields evaluate at translation and which evaluate continuously.

Appendix B

Gives instructions for the Classic Output Viewer

Glossary

Contains definitions and descriptions for many of the terms and concepts common to simulation and modeling.

Bibliography

Referenced materials and suggested readings.

Symbols and Notation

To better help you navigate this text, please review the following symbols and conventions.

Keyboard

The names of keys are displayed in capital letters. For example, ESC refers to the Escape key and CTRL refers to the Control key.

Keys are frequently specified in combinations or in a sequence of keystrokes. For example, CTRL + L means to hold down the CTRL key while pressing L. When key commands are set off by commas (e.g., ALT + N, R), press and release each of these keys (or key combinations) in the order listed. The term “arrow keys” refers collectively to the ⇧, ⇩, ⇦, and ⇨ cursor keys.

Text

Specific text you are asked to type is shown in bold type. For example, if you are directed to type **cd pmod**, you would type the lowercase letters “cd” followed by a space and the letters “pmod.”

Place holders for things such as file names and directories are shown in italics. For example, if you are directed to type *filename.mod*, enter the name of the file you wish to use (e.g., model_1.mod).

Logic

All ProModel reserved keywords are in uppercase.

Syntax example:

```
IF Start_Time = 10 THEN INC Var1
```

Product Support

Technical support is available to all licensed ProModel users with current maintenance and support agreements. Support representatives are glad to answer specific questions you may have and offer direction in solving specific modeling challenges you encounter.

Technical support

Technical support is available via: telephone, fax, ftp, and email. When you contact technical support, please be prepared to provide your user profile and a description of any problems you experience.

User Profile

- Your name and company.
- The license # found on your security key or ProModel CD.
- Hardware make and configuration.
- Network information (if applicable).
- Version number of Windows and ProModel.

Problem Description

- Brief description of the problem you are experiencing.
- What you were doing when the problem occurred.
- The exact wording of any messages that appeared on your screen.

Telephone (888) PROMODEL

Speak to a technical support engineer and resolve the problem over the phone. Our support lines are open Monday through Friday from 6:00 AM to 6:00 PM MST. The technical support number is (888) PROMODEL.

After Hours Support (801) 362-8324

In our ongoing effort to serve you better, technical support is also available after hours. Please have ready your user profile and a description of the problem you encountered.

Fax (801) 226-6046

Send a fax of the listing created when you select Print Text from the File menu along with your user profile and a description of the problem you encountered.

FTP ftp.promodel.com

For file transfers to and from ProModel Corporation via file transfer protocol, contact the ftp server at “ftp.promodel.com”. ProModel Corporation allows user access via anonymous login (your e-mail address is your password).

Email support@promodel.com

When you contact technical support via email, send your user profile and a description of the problem you encountered.

Modeling Services

If you find yourself in need of extra help or specific expertise to complete your simulation project, let us help you. ProModel Corporation Modeling Services will meet all of your needs with fast, accurate results at competitive rates.

Whether simple or complex, partial or complete, ProModel Corporation Modeling Services can create any model you require. With our vast experience in producing simulation models for many diverse applications, we are in a unique position to evaluate your system and isolate specific improvements. We work closely with you during the development process to ensure that the model we create is complete and precise.

With each simulation ProModel Corporation provides a complete, comprehensive analysis of your system. We document conclusions and results derived from the project and present you with statistics suitable for company presentations and briefings. You may even present a limited version of the model which allows repeated executions and minor revision capabilities.

For more information on modeling services, please contact:

Phone	**(801) 223-4600**
Fax	**(801) 226-6046**

Reporting Suggestions

It is our goal to make ProModel the ultimate manufacturing simulation tool. To do this, we rely on your input. Please feel free to submit comments and ideas on how we may improve the ProModel software and documentation.

Send us your comments

ProModel Product Team
ProModel Corporation
556 E. Technology Way
Orem, UT 84097

Phone	(801) 223-4600
Fax	(801) 226-6046
Email	pmteam@promodel.com

Chapter 1: Getting Started

Welcome to ProModel

ProModel is a powerful, Windows-based simulation tool for simulating and analyzing production systems of all types and sizes. ProModel provides the perfect combination of ease-of-use and complete flexibility and power for modeling nearly any situation, and its realistic animation capabilities makes simulation come to life.

ProModel provides engineers and managers the opportunity to test new ideas for system design or improvement before committing the time and resources necessary to build or alter the actual system. ProModel focuses on issues such as resource utilization, production capacity, productivity, and inventory levels. By modeling the important elements of a production system such as resource utilization, system capacity, and production schedules, you can experiment with different operating strategies and designs to achieve the best results.

As a discrete event simulator, ProModel is intended primarily for modeling discrete part manufacturing systems, although process industries can be modeled by converting bulk material into discrete units such as gallons or barrels. In addition, ProModel is designed to model systems where system events occur mainly at definite points in time. Time resolution is controllable and ranges from .01 hours to .00001 seconds.

- Typical applications for using ProModel include:
- Assembly lines• Job Shops
- Transfer lines• JIT and KANBAN systems
- Flexible Manufacturing systems• Supply chains & logistics

Use of ProModel requires only a brief orientation and virtually no programming skills. With ProModel's convenient modeling constructs and graphical user interface, model building is quick and easy. All you do is define how your particular system operates, mostly through part flow and operation logic. Automatic error and consistency checking help ensure that each model is complete prior to simulation. During simulation, an animated representation of the system appears on the screen. After the simulation, performance measures such as resource utilization, productivity and inventory levels are tabulated and may be graphed for evaluation

Where to Go from Here

1. In order to use ProModel effectively, you must understand the basics of the Microsoft Windows operating environment. If you are unfamiliar with Windows you should begin by reviewing the help information. To do this, select Help from the Start menu. If you are an accomplished Windows user, you can proceed to the next step.
2. To begin using ProModel you must install the software. This is described in Chapter 2: Installation and Registration.
3. You will want to become familiar with the basic theory of model building, and review

the model building scenario, which is covered in chapter 3: Planning the Model. If you are already familiar with modeling theory, or just want to jump into model building, skip to the next step.

4. ProModel incorporates an easy-to-use as well as intuitive user interface. Chapter 4: Modeling Environment will introduce you to the menus and options you will use to build your model.
5. Build your model. Chapters 5 and 6: Building the Model propel you into model building by describing the Build menu, which gives you access to most of the tools needed to build your model.
6. Refer to Chapter 7: Building the Logic during the model building process to learn the logical elements that will control your model during simulation.
7. Run your model. See you model come to life as its animation runs. Chapter 9: Running the Model outlines the process of running a model.
8. View, analyze, and display the data your model collected during simulation. With this information in hand you are ready to make real-world decisions to achieve your modeling goals. Chapter 10: Reports and Graphs describes the Output Viewer, which presents you with your simulation's data.

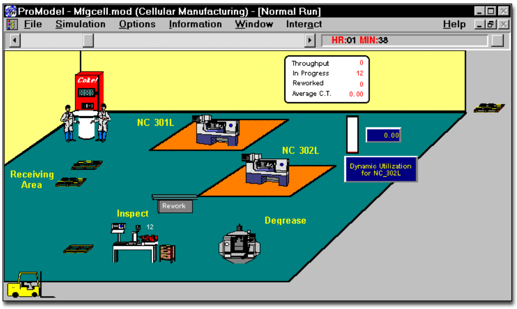
Training

Although the tutorial and documentation are both self contained, first-time users are strongly encouraged to seek formal training at some point before embarking on a complex modeling project. PROMODEL Corporation holds training courses on a regular basis for beginning and advanced users of PROMODEL simulation products. For details on course dates and times, or to register for the course nearest you, call our main office at (801) 223-4600 and ask for the ProModel Product Team.

Using ProModel

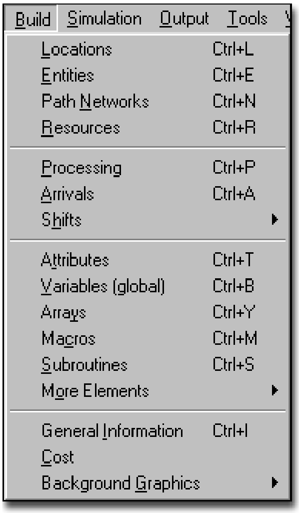
ProModel views a production system as an arrangement of processing locations, such as machines or work stations, through which parts (or entities) are processed according to some processing logic. A system may also include paths, such as aisle-ways for movement, as well as supporting resources, such as operators and material handling equipment used in the processing and movement of parts.

The example below depicts a typical manufacturing workcell with six processing locations (Receiving, NC_301L, NC_302L, Degrease, Inspect, Rework), two entity types (Pallets and Blanks), two operators, and one transporting device (forklift). Models such as this are developed rapidly and easily using ProModel's graphical user interface and point-and-click approach to modeling.



Building Models

Models are created by completing the necessary modules selected from the Build menu shown below. Each module consists of various edit tables and dialog boxes used to supply model information. A layout window also appears with graphical tools for placing locations, path networks and other modeling elements.

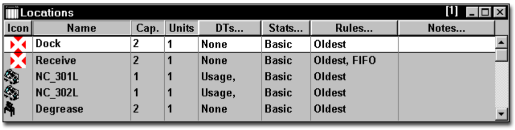


Modules may be completed in any order and with any amount of switching back and forth between modules. However, with certain exceptions, it is recommended that modules be completed in the general order in which they appear in the menu. For example, it is usually best to identify and place locations in a model, and define the entities to be processed at those locations, before actually defining processing logic. This is generally accepted since processing logic describes the flow of entities from location to location and the operations performed on, for, or in behalf entities at each location.

Edit Tables

Edit tables, such as the Locations edit table, are used extensively throughout ProModel for defining model elements. Edit tables provide direct access to model data without wading though several levels of dialog boxes. Each table consists of records, which consist of fields describing some aspect of the element. Many of these fields have heading buttons you can click (or select and press

F2) to open a dialog box for further definition of the particular element.

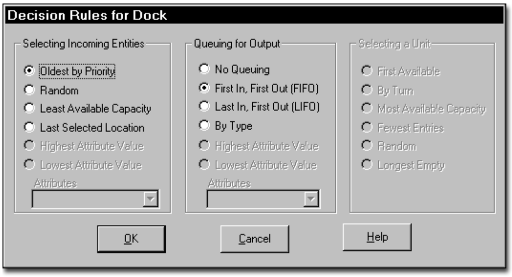


Locations

Icon	Name	Cap.	Units	DTs...	Stats...	Rules...	Notes...
	Dock	2	1	None	Basic	Oldest	
	Receive	2	1	None	Basic	Oldest, FIFO	
	NC_301L	1	1	Usage,	Basic	Oldest	
	NC_302L	1	1	Usage,	Basic	Oldest	
	Degrease	2	1	None	Basic	Oldest	

Dialog Boxes

Dialog boxes are used throughout ProModel for selecting options. The Decision Rules dialog shown below contains additional information regarding a location.



Decision Rules for Dock

Selecting Incoming Entities
- ● Oldest by Priority
- ○ Random
- ○ Least Available Capacity
- ○ Last Selected Location
- ○ Highest Attribute Value
- ○ Lowest Attribute Value

Attributes

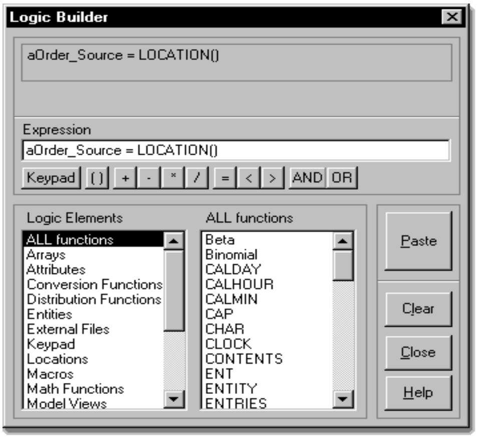
Queuing for Output
- ○ No Queuing
- ● First In, First Out (FIFO)
- ○ Last In, First Out (LIFO)
- ○ By Type
- ○ Highest Attribute Value
- ○ Lowest Attribute Value

Attributes

Selecting a Unit
- ○ First Available
- ○ By Turn
- ○ Most Available Capacity
- ○ Fewest Entries
- ○ Random
- ○ Longest Empty

Logic Builder

The Logic Builder in ProModel provides a quick and powerful way to create and insert valid statements and expressions in logic windows or fields. It takes you through the process of creating statements or expressions, as well as providing point-and-click access to every element defined in your model. The Logic Builder handles the syntax for every statement and function, allowing you to define logic by simply filling in the blanks.



Logic Builder

aOrder_Source = LOCATION()

Expression: aOrder_Source = LOCATION()

Logic Elements: ALL functions, Arrays, Attributes, Conversion Functions, Distribution Functions, Entities, External Files, Keypad, Locations, Macros, Math Functions, Model Views

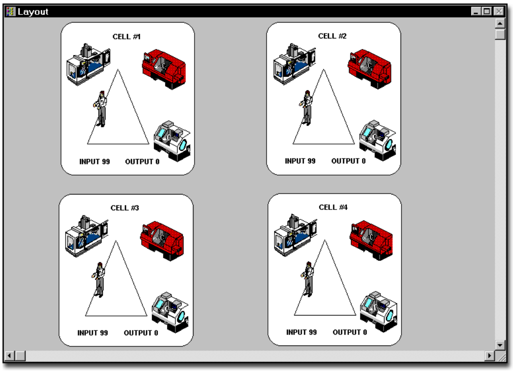
ALL functions: Beta, Binomial, CALDAY, CALHOUR, CALMIN, CAP, CHAR, CLOCK, CONTENTS, ENT, ENTITY, ENTRIES

To access the Logic Builder, click the right mouse button in any edit box which accepts an expression or anywhere inside an open logic window. You may also click the Build button on the logic window's toolbar.

Model Merging and Submodels

Another extremely useful feature in ProModel is the ability to merge two or more models into one larger model. This supports the concept of modular model building, where models are created and tested in modules, and then joined together to produce a final model.

The use of sub-models further allows you to build common model elements such as work cells and place them repeatedly in a model. The elements of each sub-model (i.e., locations, entities, resources, etc.) may all be given a common prefix or suffix to differentiate them from the same elements of another sub-model.



For more information on merging models, see “Model Merging” on page 70.

On-Line Help

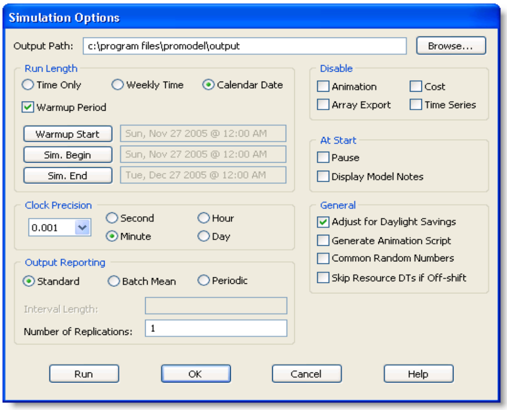
Regardless of where you are in the model building process, the on-line help system is available to provide context specific definitions and instructions. The help system can even run in the background, allowing instant access to help whenever you need it.

Like all Windows applications, you can locate the help you need quickly using the standard help tools.

Running Models

Completed models are run using the Simulation menu. Model data is automatically checked for consistency and completeness before each simulation begins.

As shown in the dialog below, ProModel allows you to specify a run length, warm-up period, number of replications, and other special options before running a simulation. In addition, you may run the models with or without animation. For more information on running models, see “Running the Model” on page 347.


Simulation Options

Output Path: c:\program files\promodel\output

Run Length
○ Time Only ○ Weekly Time ● Calendar Date
☑ Warmup Period
Warmup Start: Sun, Nov 27 2005 @ 12:00 AM
Sim. Begin: Sun, Nov 27 2005 @ 12:00 AM
Sim. End: Tue, Dec 27 2005 @ 12:00 AM

Clock Precision
0.001
○ Second ○ Hour
● Minute ○ Day

Output Reporting
● Standard ○ Batch Mean ○ Periodic
Interval Length:
Number of Replications: 1

Disable
□ Animation □ Cost
□ Array Export □ Time Series

At Start
□ Pause
□ Display Model Notes

General
☑ Adjust for Daylight Savings
□ Generate Animation Script
□ Common Random Numbers
□ Skip Resource DTs if Off-shift

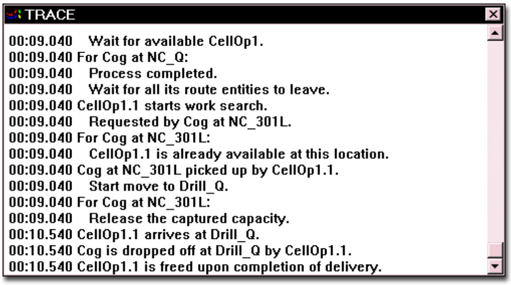
Creating Run-Time Models

One of the most powerful features in ProModel is the ability to create models that may be run by others who do not own the full software. PROMODEL allows you to install the software on any machine to run a model. However, without the security key, the user has limited ability to build or make changes to a model.

You may wish to develop run-time models to allow the user to perform "what-if" analyses with the models. PROMODEL encourages you to develop demonstration models, and allows you to distribute them freely. These models can be run by installing ProModel from the CD-Rom onto a computer without a security key.

Trace Window

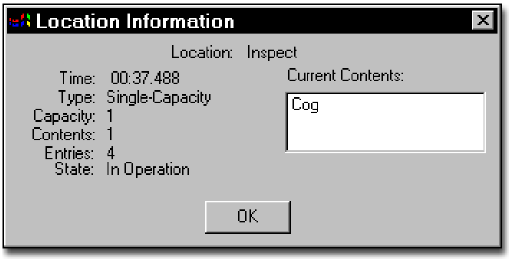
Several tools are available to help you verify and validate your models. During model execution you may trace the activity of the simulation events to see exactly what is happening in the model. The example below shows a brief portion of a trace window.



```
00:09.040   Wait for available CellOp1.
00:09.040 For Cog at NC_Q:
00:09.040   Process completed.
00:09.040   Wait for all its route entities to leave.
00:09.040 CellOp1.1 starts work search.
00:09.040   Requested by Cog at NC_301L.
00:09.040 For Cog at NC_301L:
00:09.040   CellOp1.1 is already available at this location.
00:09.040 Cog at NC_301L picked up by CellOp1.1.
00:09.040   Start move to Drill_Q.
00:09.040 For Cog at NC_301L:
00:09.040   Release the captured capacity.
00:10.540 CellOp1.1 arrives at Drill_Q.
00:10.540 Cog is dropped off at Drill_Q by CellOp1.1.
00:10.540 CellOp1.1 is freed upon completion of delivery.
```

Location Information Windows

In addition to trace files, you may track the activity at any location through a Location Information window. This feature gives up-to-the-minute information about a selected location, such as current contents, total entries, and operational state (i.e., idle, blocked, or in operation).



Location: Inspect

Time: 00:37.488
Type: Single-Capacity
Capacity: 1
Contents: 1
Entries: 4
State: In Operation

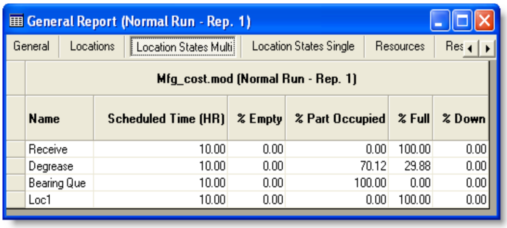
Current Contents:
Cog

Viewing Output

The output generator gathers statistics on each location, entity, resource, path network, and variable in the system. You may, however, turn off reporting capability for any element you do not wish to include. The default level of statistics is at the summary level (i.e., average values, % values, and final values), although detailed history plots can be gathered on such things as utilization, queue fluctuations, and variable values. For more information about viewing the results of a simulation, see “Reports and Graphs” on page 373.

Simulation results may be presented in either tabular or graphic format, including histograms, pie charts, plots and bar graphs. Multiple output results can even be compared on the same chart.

The example below shows a portion of a general report for a model.

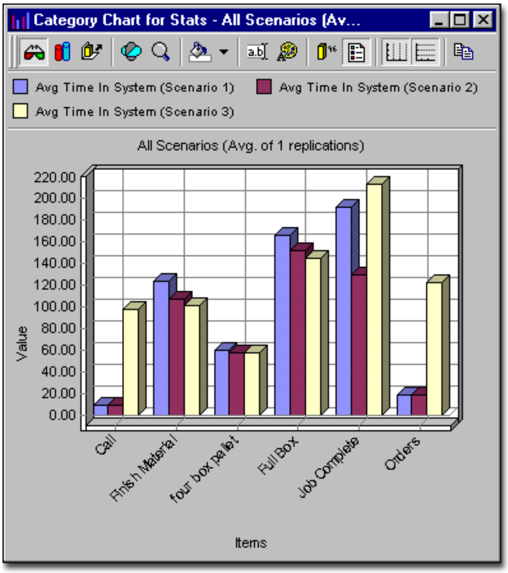


Mfg_cost.mod (Normal Run - Rep. 1)

Name	Scheduled Time (HR)	% Empty	% Part Occupied	% Full	% Down
Receive	10.00	0.00	0.00	100.00	0.00
Degrease	10.00	0.00	70.12	29.88	0.00
Bearing Que	10.00	0.00	100.00	0.00	0.00
Loc1	10.00	0.00	0.00	100.00	0.00

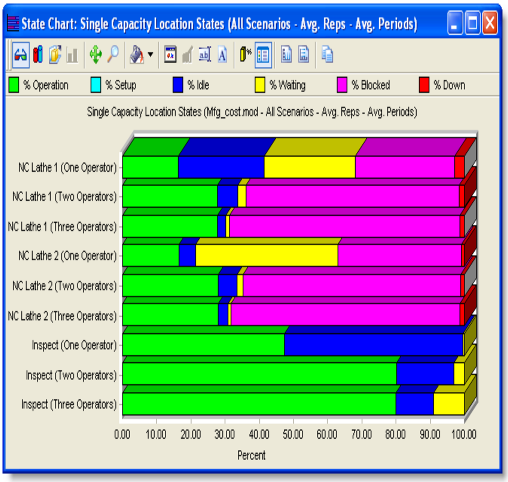
Graphical Analysis

In addition to the general report shown above, almost all report elements of a model may be displayed graphically. Below is an example of a Category Chart.



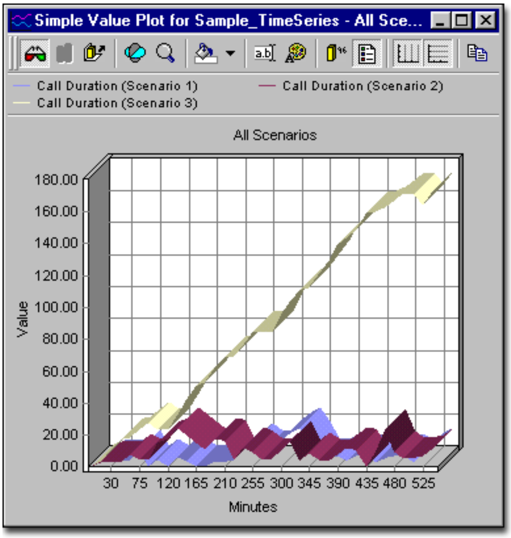
Utilization Graphs

Quickly generate utilization graphs of various types such as the State Chart shown below.


State Chart: Single Capacity Location States (All Scenarios - Avg. Reps - Avg. Periods)
🟩 % Operation
🟦 % Setup
🟦 % Idle
🟨 % Waiting
🟪 % Blocked
🟥 % Down
Single Capacity Location States (Mfg_cost.mod - All Scenarios - Avg. Reps - Avg. Periods)
NC Lathe 1 (One Operator)
NC Lathe 1 (Two Operators)
NC Lathe 1 (Three Operators)
NC Lathe 2 (One Operator)
NC Lathe 2 (Two Operators)
NC Lathe 2 (Three Operators)
Inspect (One Operator)
Inspect (Two Operators)
Inspect (Three Operators)
0.00
10.00
20.00
30.00
40.00
50.00
60.00
70.00
80.00
90.00
100.00
Percent

Timeplots

In addition to Category and State Charts, Timeplot Charts, which show variable values as they change over time, can be created.


Simple Value Plot for Sample_TimeSeries - All Sce...
Call Duration (Scenario 1)
Call Duration (Scenario 2)
Call Duration (Scenario 3)
All Scenarios
180.00
160.00
140.00
120.00
100.00
80.00
60.00
40.00
20.00
0.00
Value
30
75
120
165
210
255
300
345
390
435
480
525
Minutes

Chapter 2: Installation and Registration

General Setup Information

This section describes how to install and register ProModel.

Hardware Requirements

Minimum

Pentium 3 - 800 or better
512MB RAM
200 MB Free Disk Space
SVGA Monitor (1024 x 768)
CD ROM
Mouse
Windows 98 SE or higher

Recommended

Pentium 4 or better
1 GB RAM
1 GB Free Disk Space
SVGA Monitor (1280 x 1024 x 32 bit color)
CD ROM
Sound card
Internet access
Mouse
Windows 2000 or Windows XP

Installation Procedure for a Stand-alone PC

The installation program will decompress and copy files from the CD-ROM into user-specified directories.

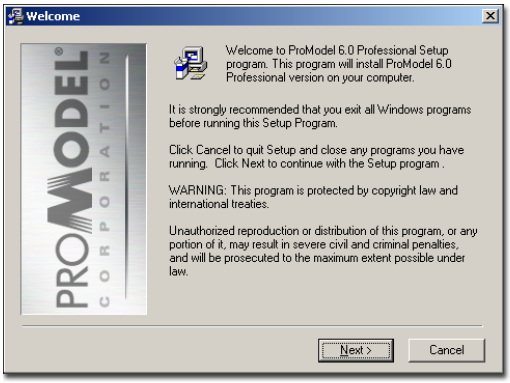
Technical Support

If you encounter problems during installation, please call the Product Team at (888) PRO-MODEL between 6:00 AM and 6:00 PM MST, Monday through Friday. We will be glad to help you get up and running.

How to install ProModel:

1. Start Windows.
2. Insert the CD-ROM.

3. The ProModel Setup program will open automatically, and display the following window.



Welcome to ProModel 6.0 Professional Setup program. This program will install ProModel 6.0 Professional version on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup Program.

Click Cancel to quit Setup and close any programs you have running. Click Next to continue with the Setup program .

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

Please note

If the Setup Program did not open automatically, select Run... from the Windows Start menu. Type x:\install.exe (where x is the CD-ROM drive letter) and press ENTER. The proper dialog will the appear

4. Click **Next** to proceed with the installation

5. Review the License Agreement. If you wish to accept the agreement and continue with the installation, , select the button to accept the agreement, and then click **Next**.

6. Select Standard Package and click **Next.**

7. From the dialog that appears, choose the components that you would like the Setup program to install. It is recommended that you allow the Setup program to install all components, hard drive space permitting. When you have finished selecting your components, click **Next**.

8. Choose the destination directory for the install. If you wish to change the destination from the default, select the **Browse** button and choose a new directory. When you have chosen the destination directory, click **Next.**

9. The next dialog allows you to have the Setup program create backups of any files that might be replaced during the installation. This is helpful if you are installing this version of the ProModel software on a computer that already has ProModel installed on it. If hard drive space permits, it is recommended that you allow the Setup program to backup previous files. When you are ready to continue with the installation, click **Next.**

10. The Setup program will add a new Windows Program group containing ProModel program icons to the Program Manager. When you have selected where you want the Program icons to appear, click **Next.**

11. The Setup program is now ready to install ProModel on your computer. If you wish to make changes to the options you have previously selected, click the **Back** button to return to any point in the installation you wish. Otherwise, click **Next** to allow the Setup program to install ProModel.

The Setup program may require you to reboot your computer during the setup. If you do so, the Setup program will automatically launch after the reboot, and the installation will continue.

When you have completed the installation, you can run ProModel from the Windows Start menu.

Upon running ProModel for the first time, you will be prompted to register ProModel. For instruction on doing so, see the section at the end of this chapter on registering ProModel.

Installation Procedure for Network Version

In a multi-user environment, license tracking with multiple keys is difficult. To combat this, PROMODEL Corporation offers a network version of ProModel that allows a single machine to control license usage (via a network connection) for several users. To use this method of license tracking, you must install a *network* version of ProModel. The network version of ProModel includes a hardware security key and the license manager software you will use to control license usage at your site.

Overview

When you install and run the network version of ProModel from a workstation's local hard drive or a file server share, the workstation searches your network for a license key server rather than check its own printer port for a security key. Once it locates the license key server, the workstation asks the license manager software (running on the key server) if a license is available. If a license is available, the license manager software will assign a license to the workstation. When you exit ProModel, the license you used becomes available to another user.

Check the support web page for periodic updates to the installation information, software configuration, and the license server software for ProModel.

http://www.promodel.com/support

Please note

If you are running a network version of ProModel on a routed network and your license key server and workstations are located on different sub-nets, see "Find a License Key Server on a Routed Network" on page 23.

Installation Types

Local Machine Install

This procedure allows you to install to and execute ProModel from the local hard drive of a workstation, yet still require the workstation to make a license request from the license server.

Pros

- Locally stored files minimize start-up time.
- One-step installation as opposed to a file server install.
- Limits network usage to short license requests only and allows users to connect remotely due to lower network traffic needs.

Cons

- You must install ProModel from the CD at each workstation.
- Install occupies more hard drive space on each workstation than when you install it on a file server share.

File Server Install

This procedure allows you to install ProModel to a share on a file server. From each workstation that will run ProModel, run the workstation setup program, pmwsetp.exe (this will create a program group, icons, and copy several files to the windows directory).

Pros

- Saves hard drive space on the workstation by running ProModel from the file server.

- End-user can run workstation setup program without administrative help.

Cons

- File server must always be available.

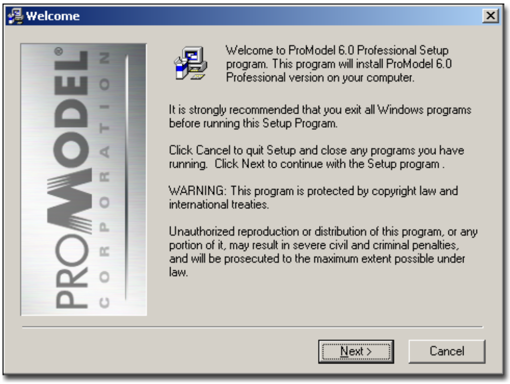
Set Up Network License Server

Whether you choose to do a file server share or local hard drive install of the network version of ProModel, you must set up a license server. The license server consists of the security key that plugs into the license server's printer port, and the license manager software. When installed, the license server responds to ProModel license requests and allows workstations to run either a full version (if sufficient licenses are available) or a limited run-time version.

Local Machine Install

Install Program Files on Local Machine:

1. Start Windows.
2. Insert the CD-ROM.
3. The ProModel Setup program will open automatically, and display the following window.



Welcome

Welcome to ProModel 6.0 Professional Setup program. This program will install ProModel 6.0 Professional version on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup Program.

Click Cancel to quit Setup and close any programs you have running. Click Next to continue with the Setup program .

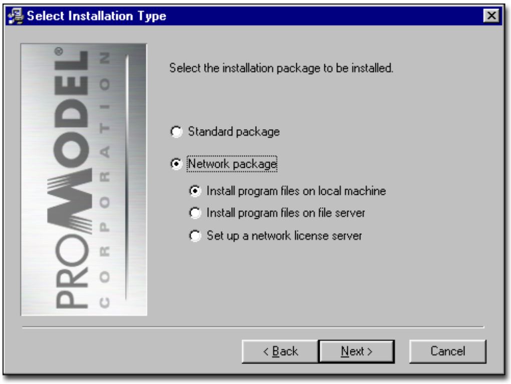
WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

Please note

If the Setup Program did not open automatically, select Run... from the Windows Start menu. Type x:\install.exe (where x is the CD-ROM drive letter) and press ENTER. The proper dialog will the appear

4. Click **Next** to proceed with the installation
5. Review the License Agreement. If you wish to accept the agreement and continue with the installation, , select the button to accept the agreement, and then click **Next**.
6. From the dialog that appears, select **Network Package**. The dialog will then appear as below:



Select Installation Type

Select the installation package to be installed.

○ Standard package

● Network package

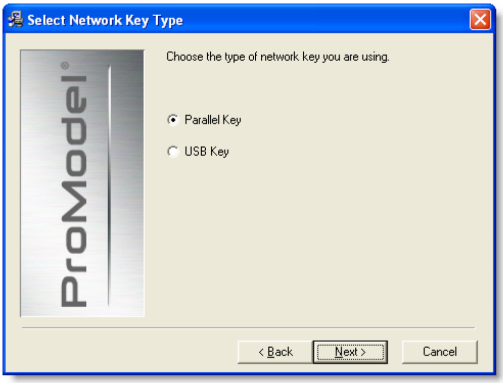
● Install program files on local machine

○ Install program files on file server

○ Set up a network license server

7. Select **Install program files on local machine**, and click **Next**.

8. Select the type of hardware key you are using on the network server, and click **Next**.


Select Network Key Type

Choose the type of network key you are using.

● Parallel Key

○ USB Key

9. From the dialog that appears, choose the components that you would like the Setup program to install. It is recommended that you allow the Setup program to install all components, hard drive space permitting. When you have finished selecting your components, click **Next**.

10. Choose the destination directory for the install. If you wish to change the destination from the default, select the **Browse** button and choose a new directory. When you have chosen the destination directory, click **Next.**

11. The next dialog allows you to have the Setup program create backups of any files that might be replaced during the installation. This is helpful if you are installing this version of the ProModel software on a computer that already has ProModel installed on it. Later, if you choose, you can have the installation rolled back and the original files restored. If hard drive space permits, it is recommended that you allow the Setup program to backup previous files. When you are ready to continue with the installation, click **Next.**

12. The Setup program will add a new Windows Program group containing ProModel program icons to the Program Manager.

When you have selected where you want the Program icons to appear, click **Next.**

13. The Setup program is now ready to install ProModel on your computer. If you wish to make changes to the options you have previously selected, click the **Back** button to return to any point in the installation you wish. Otherwise, click **Next** to allow the Setup program to install ProModel.

The Setup program may require you to reboot your computer during the setup. If you do so, the Setup program will automatically launch after the reboot, and the installation will continue.

When you have completed the installation, you must set up a *license server.* See “Set Up License Server” on page 21, for information on how to do so.

Please note

If you are running a network version of ProModel on a routed network and your license key server and workstations are located on different sub-nets, see “Find a License Key Server on a Routed Network” on page 23.

File Server Install

Install Programs & Grant Rights/ Permissions

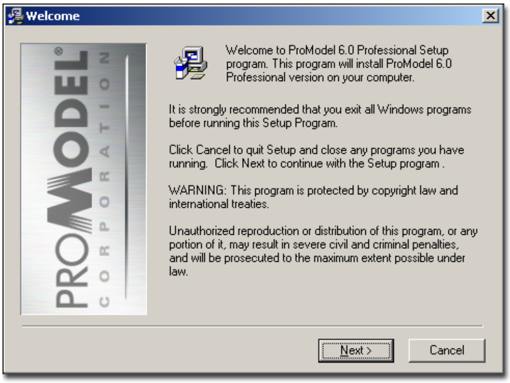
From a PC on the network, install the software on the file server and grant appropriate rights. Users will need at least the equivalent rights of read and file scan for the PROMODEL directory.

Install Program Files on File Server:

1. Start Windows.

2. Insert the CD-ROM.

3. The ProModel Setup program will open automatically, and display the following window.



Welcome

Welcome to ProModel 6.0 Professional Setup program. This program will install ProModel 6.0 Professional version on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup Program.

Click Cancel to quit Setup and close any programs you have running. Click Next to continue with the Setup program .

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

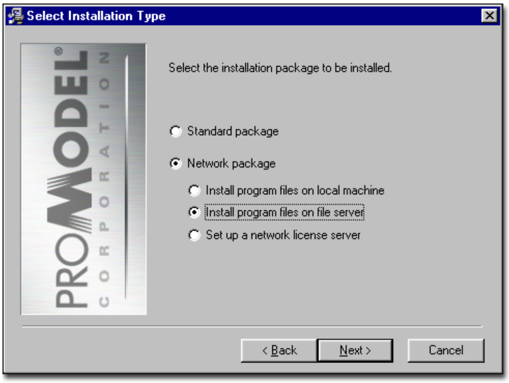
Please note

If the Setup Program did not open automatically, select Run... from the Windows Start menu. Type x:\install.exe (where x is the CD-ROM drive letter) and press ENTER. The proper dialog will the appear

4. Click **Next** to proceed with the installation

5. Review the License Agreement. If you wish to accept the agreement and continue with the installation, , select the button to accept the agreement, and then click **Next**.

6. From the dialog that appears, select **Network Package**. The dialog will then appear as below:



Select Installation Type

Select the installation package to be installed.

○ Standard package

● Network package

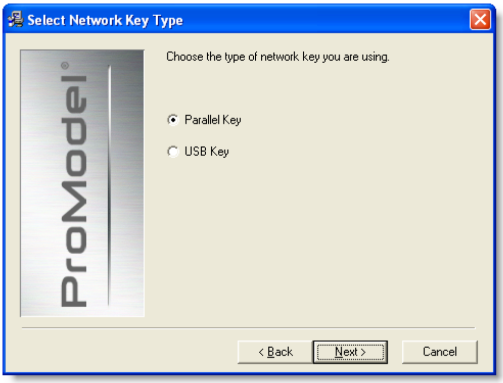
○ Install program files on local machine

● Install program files on file server

○ Set up a network license server

7. Select **Install program files on file server**, and click **Next**.

8. Select the type of hardware key you are using on the network server, and click **Next**.



Select Network Key Type

Choose the type of network key you are using.

● Parallel Key

○ USB Key

9. From the dialog that appears, choose the components that you would like the Setup program to install. It is recommended that you allow the Setup program to install all components, hard drive space permitting. When you have finished selecting your components, click **Next**.

10. Choose the destination directory for the install. If you wish to change the destination from the default, select the **Browse** button and choose a new directory. When you have chosen the destination directory, click **Next.**

11. The next dialog allows you to have the Setup program create backups of any files that might be replaced during the installation. This is helpful if you are installing this version of the ProModel software on a computer that already has ProModel installed on it. Later, if you choose, you can have the installation rolled back and the original files restored. If hard drive space permits, it is recommended that you allow the Setup program to backup previous files. When you are ready to continue with the installation, click **Next.**

12. The Setup program will add a new Windows Program group containing ProModel program icons to the Program Manager. When you have selected where you want the Program icons to appear, click **Next.**

13. The Setup program is now ready to install ProModel on your server. If you wish to make changes to the options you have previously selected, click the **Back** button to return to any point in the installation you wish. Otherwise, click **Next** to allow the Setup program to install ProModel.

The Setup program may require you to reboot your computer during the setup. If you do so, the Setup program will automatically launch after the reboot, and the installation will continue.

When you have completed the installation, you can setup each workstation you will use to run ProModel. See "Workstation Set up," which is the next section, for information on setting up multiple workstations.

When you have completed setting up workstations, you must set up a *license server*. See “Set Up License Server” on page 21, for information on how to do so.

Workstation Set up

Set up a Workstation:

1. Run **ProModel Workstation Setup.exeS** (located in the directory to which you installed ProModel on the file server) from the workstation you will use to run ProModel.

2. Click **Next**.

3. Select the components you wish to install and click **Next**.

4. Select the folder to which you will to install ProModel and click **Next**.

5. Select the program group you wish to use and click **Next**. (By default, ProModel creates its own program group).

6. Click **Next** to finish installing ProModel.

Please note

If you are running a network version of ProModel on a routed network and your license key server and workstations are located on different sub-nets, see “Find a License Key Server on a Routed Network” on page 23.

Set Up License Server

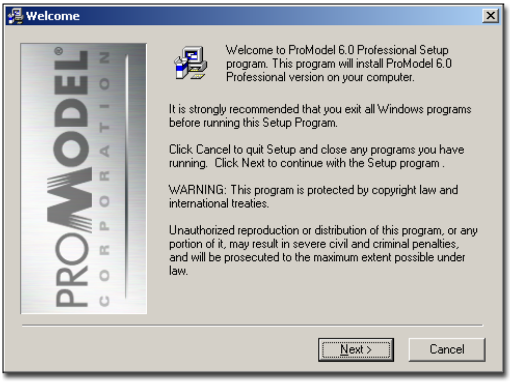
PROMODEL software monitors licensed user activity through a security key attached to the parallel port (for the network version of ProModel, you must have the red key supplied by PROMODEL Corporation). The machine to which you attach the key becomes the license manager—whether it is a dedicated file server or a workstation. The security server can run on Windows 95, 98, 2000, or NT 4.0 and listen for incoming license requests over any of the following network protocols: IPX, TCP/IP, and NetBEUI. If the security server is not functioning, ProModel will start as a limited run-time version.

Please note

If you are running a network version of ProModel on a routed network and your license key server and workstations are located on different sub-nets, see “Find a License Key Server on a Routed Network” on page 23.

Set Up License Server:

1. Start Windows.
2. Insert the CD-ROM.
3. The ProModel Setup program will open automatically, and display the following window.



Welcome

Welcome to ProModel 6.0 Professional Setup program. This program will install ProModel 6.0 Professional version on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup Program.

Click Cancel to quit Setup and close any programs you have running. Click Next to continue with the Setup program .

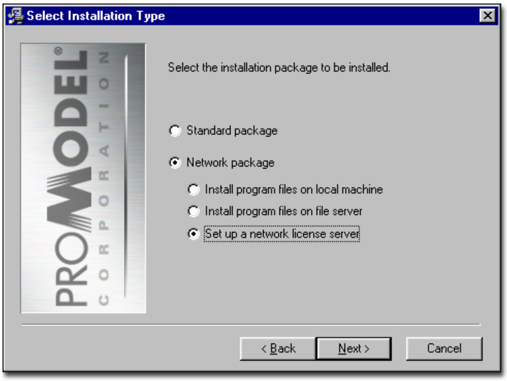
WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

Please note

If the Setup Program did not open automatically, select Run... from the Windows Start menu. Type x:\install.exe (where x is the CD-ROM drive letter) and press ENTER. The proper dialog will the appear

4. Click **Next** to proceed with the installation
5. Review the License Agreement. If you wish to accept the agreement and continue with the installation, click **Next**.
6. From the dialog that appears, select **Network Package**. The dialog will then appear as below:

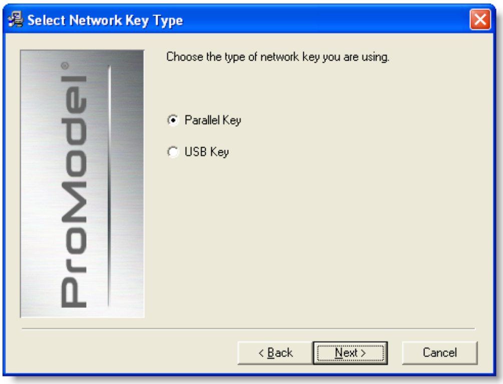


Select Installation Type

Select the installation package to be installed.

○ Standard package
● Network package
  ○ Install program files on local machine
  ○ Install program files on file server
  ● Set up a network license server

7. Select **Set up a network license server**, and click **Next**.
8. Select the type of hardware key you are using on the network server, and click **Next**.



Select Network Key Type

Choose the type of network key you are using.

● Parallel Key
○ USB Key

9. The Install Wizard for the NetHASP License Manager will appear. Click **Yes**, and follow the on-screen directions to install the manager. If you click **No**, you will have to set up the license server manually, which is described in the next section.

The Setup program may require you to reboot your computer during the setup. If you do so, the Setup program will automatically launch after the reboot, and the installation will continue.

You need to install the device driver only once. However, you must launch the license manager program *each time you reboot the key server machine*. To automatically start the license manager each time you reboot the system, create and place an icon for NHSRVW32.EXE in the workstation's startup folder.

When you have completed installing either you local machine or server setup, and finished installing the License Server, you are ready to run ProModel.

Set Up License Server Manually

After successfully running the setup for the license server, if you chose to manually install the device driver for the security key and launch the license manager software, follow the steps below.

Install the Device Driver:

1. Open a DOS box and run **HINSTALL -i** from the directory to which you installed the license server.

2. After you receive a confirmation that the driver installed successfully, reboot the workstation

Start the License Manager Software:

1. From the directory to which you installed the license server, run NHSRVW32.EXE. A window will appear and show the network protocols to which the license manager will listen for license requests.

You need to install the device driver only once. However, you must launch the license manager program each time you reboot the key server. To automatically start the license manager each time you reboot the system, create and place an icon for NHSRVW32.EXE in the workstation's startup folder.

Find a License Key Server on a Routed Network

In order for a workstation to run the network version of ProModel properly, the workstation must check out a license from a license server. To make a license request, the workstation sends a broadcast message out on the network and awaits a response from the license server. If the license server and the workstation running ProModel are on the same sub net of a routed network (or on the same network of a non-routed network), the license server receives the request and responds. Due to the nature of routed networks, if the license server and workstation running ProModel are not on the same sub net of a routed network, the key server will not receive the license request broadcast.

To resolve this problem, the workstation must send a license request directly to the computer on the network set up as the license server. To identify which computer on the network is the license server, you must create and store a text file called NETHASP.INI in the directory to which you installed ProModel on the workstation or file server.

For TCPIP-based network the NETHASP.INI file must contain the following lines

```
[NH_COMMON]
NH_TCPIP=Enabled;


[NH_TCPIP]
NH_TCPIP_METHOD=TCP
NH_SERVER_ADDR=<Enter the license server's IP address here>
```

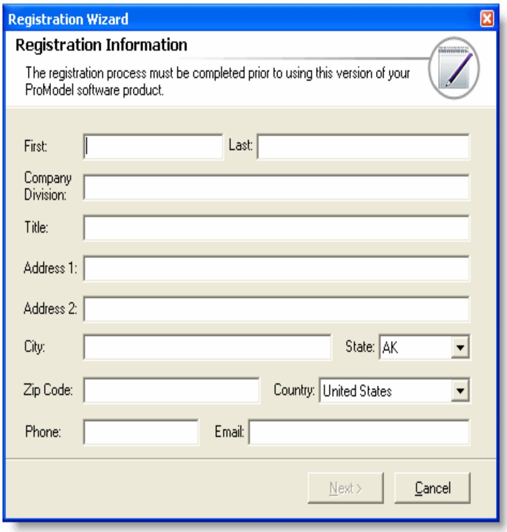
Registering ProModel

Your ProModel product must be registered. If you do not register ProModel, you will only have access to a limited, evaluation version of ProModel. This version will not allow expansive model building.

The registration process for ProModel depends on the type of installation you chose.

Registration for a Stand-alone PC Installation

When you run ProModel for the first time after installing it on a Stand-alone PC, you will be prompted to register ProModel. The following screen will appear.



Registration Wizard

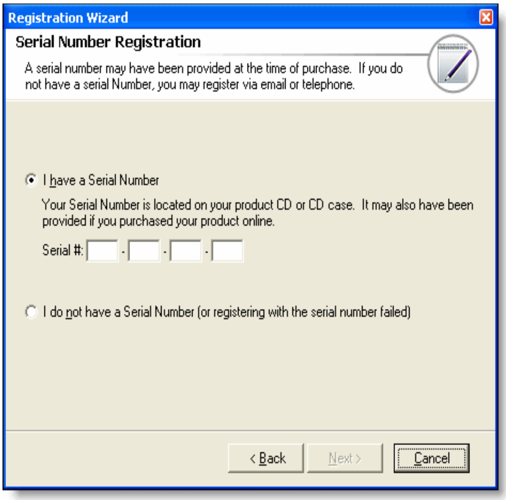
Registration Information

The registration process must be completed prior to using this version of your ProModel software product.

State: AK

Country: United States

When you have entered your information and clicked next, you will be prompted to enter your Serial Number.



Registration Wizard

Serial Number Registration

A serial number may have been provided at the time of purchase. If you do not have a serial Number, you may register via email or telephone.

● I have a Serial Number

Your Serial Number is located on your product CD or CD case. It may also have been provided if you purchased your product online.

Serial #: ___ - ___ - ___ - ___

○ I do not have a Serial Number (or registering with the serial number failed)

Depending on your organization's software licensing arrangement, a Serial Number may be included with your CD or possibly obtained by contacting the ProModel Corporation licensing representative for your organization.

If you do not have a Serial Number, you may still register using the instructions "How to Register without a Serial Number" on page 25.

How to Register with a Serial Number

1. When you have obtained a valid Serial Number, enter it into the registration dialog, and then click the **Next** button.

2. If a connection cannot be made to the ProModel Corporation server, an Internet connectivity issue (firewall, etc.) may exist. If this is the case, you will be prompted that the registration was not successful, in which case you may still register without a serial number.

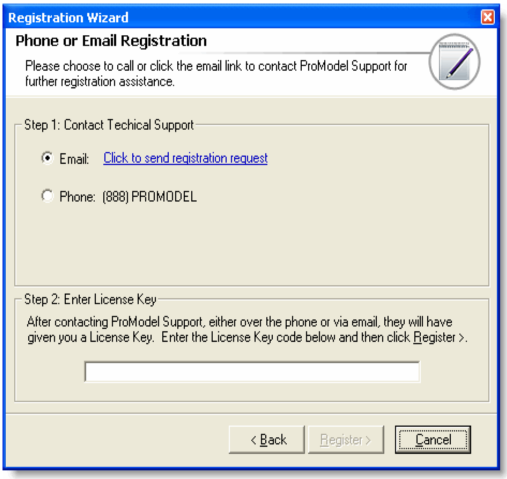
See "How to Register without a Serial Number" on page 25.

3. When the serial number has been successfully sent to the ProModel Corporation server, you will be given a ProModel username and password. Record this information, since you will use it to obtain updates to ProModel in the future.

4. Check the "I have recorded this information for future use" box, and then click Exit to finish the registration.

How to Register without a Serial Number

1. If you do not have a serial number, or your serial number cannot be used due to Internet connectivity issues, choose the "I do not have a Serial Number (or registering with the serial number failed)" option, and click Next.



Registration Wizard

Phone or Email Registration

Please choose to call or click the email link to contact ProModel Support for further registration assistance.

Step 1: Contact Techical Support

● Email: Click to send registration request

○ Phone: (888) PROMODEL

Step 2: Enter License Key

After contacting ProModel Support, either over the phone or via email, they will have given you a License Key. Enter the License Key code below and then click Register >.

2. You may choose to email (this step) or phone (next step) ProModel Support to register your product. If you would like to email ProModel Support, follow the "Click to send registration request" link that is next to the Email option.

An email message will be composed using your computer's default email application, and you can then choose to send it.

While waiting for a response from ProModel Support, you may leave the Registration dialog open, or close it until you receive your response.

The reply you receive from ProModel Support will contain a License Key code. If you have closed ProModel, relaunch it, and return to the Phone or Email Registration window. In the "Step 2: Enter License Key" area, enter the License Key code that is in the email reply from ProModel Support.

You may then click "Register" to complete the registration process.

3. If you would like to register over the phone, call ProModel Support at (888) PROMODEL. A ProModel Support representative will guide you through the rest of registration process.

You will now have access to the full version of ProModel.

If you choose not to register ProModel at this time, click **Cancel** without entering registration information. You will then have access to the limited, evaluation version of ProModel.

Registration for a Network Installation

If you purchased a network version of ProModel, you will have received a hardware key with your ProModel software. To register your network version of ProModel, simply attach the hardware key to your computer's LPT1 port.

Transferring Your Software Key

As you use ProModel it may become necessary to transfer your software key from one PC to another. Since you may only have one installation of ProModel for every software key, or license, you have purchase, PROMODEL provides you with a utility program, LicenseManager, to move your license, while keeping your software key valid.

The LicenseManager is a stand alone program, which can be accessed through the Windows Start menu.

Moving a License

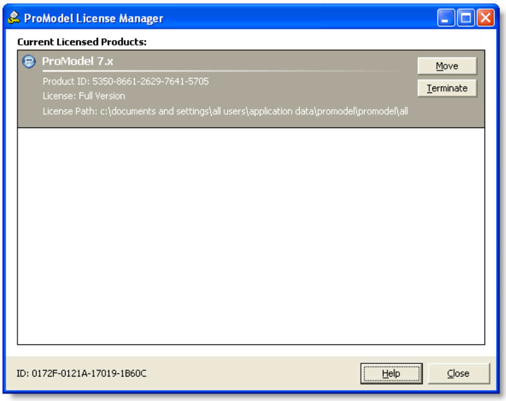
A license may be moved from one computer to another by moving its software key. When a software key is moved using the LicenseManager, the original computer will no longer run the full version of ProModel.

In preparation of moving a license to another computer, be sure to install ProModel on the computer to which you want to move the license.

How to move a license

1. Launch the LicenseManager from the Windows Start menu.

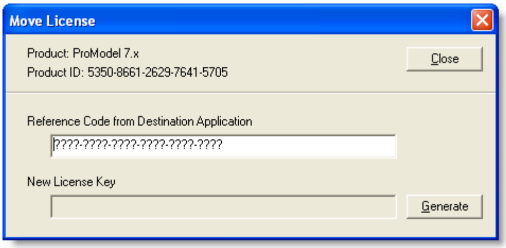
2. The following dialog will appear, showing you the licenses you currently have on your machine.



Product ID: 5350-8661-2629-7641-5705
License: Full Version
License Path: c:\documents and settings\all users\application data\promodel\promodel\all

ID: 0172F-0121A-17019-1B60C

3. If you have multiple licenses, click on the name of the license you want to move. Click **Move**. The following dialog will appear.



Product: ProModel 7.x
Product ID: 5350-8661-2629-7641-5705

Reference Code from Destination Application
????-????-????-????-????-????

New License Key

4. Enter the reference code. This will be the code generated by the computer you are moving a license to. The reference code will be displayed when you attempt to register ProModel on the machine you are moving the license to.

5. WARNING: Proceeding with the next step will permanently remove your software key from the computer; however, it is necessary to complete the next step to move the software key to the new machine.

6. After entering the reference code, click **Generate**. A new license key will be displayed. You software key on the original computer has now been removed.

7. Copy the new license key into the license key field of the registration dialog on the machine you are moving the license to in order to complete its registration.

8. The full version of ProModel will now run on new machine. The original computer will no longer run the full version of ProModel, but the evaluation version instead.

Terminating a License

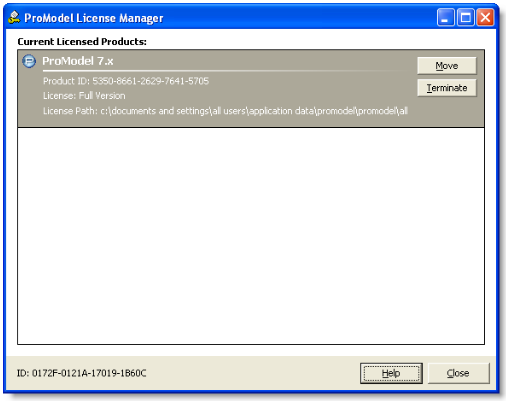
Terminating a license permanently removes the valid software key from a machine running ProModel. Terminating a license may be necessary to receive a new software key from PROMODEL.

When you terminate a license, the LicenseManager will give you a termination code, which is your proof to PROMODEL that you have permanently removed the software key from you computer. Only then can PROMODEL issue you a new software key for any current license you may have.

How to terminate a license

1. Launch the LicenseManager from the Windows Start menu.

2. The following dialog will appear, showing you the licenses you currently have on your machine.

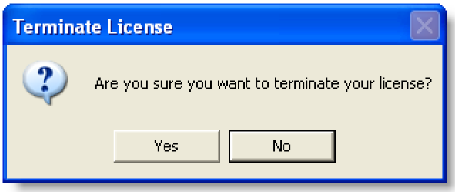


Current Licensed Products:

ProModel 7.x
Product ID: 5350-8661-2629-7641-5705
License: Full Version
License Path: c:\documents and settings\all users\application data\promodel\promodel\all

ID: 0172F-0121A-17019-1B60C

3. If you have multiple licenses, click on the name of the license you want to terminate. Click **Terminate**. The following dialog will appear.

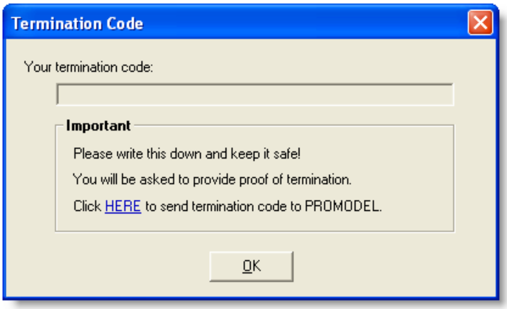


Are you sure you want to terminate your license?

4. Click **Yes**.

5. WARNING: Completing the next step will permanently remove your software key from your computer.

6. You will once again be prompted to continue with the termination. Click **OK**. Your software key will be terminated, and the following dialog will appear.



Termination Code

Your termination code:

Important

Please write this down and keep it safe!

You will be asked to provide proof of termination.

Click HERE to send termination code to PROMODEL.

7. Write down and save your termination code. This is your proof that your have terminated your software key. Click the **HERE** link to E-mail the code to PROMODEL

8. Click OK to complete the termination process. Your installed version of ProModel will now run in evaluation mode.

Checking for ProModel Updates

The Check for ProModel Update feature checks the PROMODEL Web server for an update to ProModel, downloads any available update, and then installs the update. In order to have access to updates, you will need a PROMODEL Solutions Cafe username and password, and a current Maintenance and Support agreement. If you have questions regards your M&S agreement, or username and password, please contact PROMODEL Technical Support at support@promodel.com.

There are two ways to launch the Check for Update client:

- Upon completion of the main ProModel 6.0 professional install, you will have the option to check for updates by selecting the "Check for updates to ProModel" option on the final installation screen.
- Or, you may launch the Update client by selecting "Check for ProModel Update" from the Start Menu’s PROMODEL Icon group.

It is recommend that you check regularly for updates to ProModel.

Software License Key FAQ

1. **I need to move my software license key to another computer, can I do this?**
   Yes. Using the License Manager, which can be found in the Start menu within the ProModel 6.0/PowerTools group, you may move a license. It is recommended you use the Move option in the License Manager to move a license between computers. A description of the License Manager and its functionality can be found in chapter 2 of the User Guide.
2. **I want to upgrade my computer’s operating system, or make a major change to my computer’s configuration, should I be concerned?**
   The software license key is sensitive to changes to your computer’s configuration. This is to prevent the key’s unauthorized duplication to multiple computers. If you plan on making major changes to your computer’s configuration, you must take one of the following steps to protect your software license key:
   - Move your key to another computer, using the License Manager. When you have completed the changes to the original computer, you may then move the license key back.

- Or, if moving the key to another computer is not an option, you may terminate your software license key using the License Manager. Doing so will create a Termination Code, which you will use as proof of the key's removal when you contact PROMODEL Support (1-888-PROMODEL) for a new license key after you have completed the changes to your computer.

3. **My software license key no longer appears to be functioning. What should I do?**
   Your software license key may have been disturbed by changes to your computer's configuration. Reboot your computer in an attempt to reset the configuration to a point that is recognizable by the license key. If the problem persists, please contact PROMODEL Support (1-888-PROMODEL).

Chapter 3: Planning the Model

Steps for Doing Simulation

Introduction

Doing simulation requires more than just knowing how to use a simulation product. A simulation study is, by its very nature, a project. Like any project, there are tasks to be completed and resources required to complete them. To be successful, a simulation project should be planned with an understanding of the requirements of each of the tasks involved. Many failures result from hastily jumping into a simulation without first taking time to consider the steps involved and developing a plan for proceeding.

Simulation modeling requires good analytical, statistical, communication, organizational, and engineering skills. The modeler must understand the system being investigated and be able to sort through complex cause-and-effect relationships that determine system performance. At least a basic foundation in statistics is needed to properly design experiments and correctly analyze and interpret input and output data. Ongoing communication with owners, stakeholders, and customers during a simulation study is also vital to ensure that a purposeful model is built and that everyone understands the objectives, assumptions, and results of the study.

General Procedure

A decision to do a simulation usually results from a perception that simulation can help resolve one or more issues associated with the design of a new system or the modification of an existing system. Before launching into a simulation project, one or more individuals should have been assigned to the study who have at least a basic knowledge of the system to be studied and the issues of concern. Enough background information should have been obtained about the nature of the problem to determine whether simulation is a suitable solution. If the simulation is being conducted by individuals inside the company, there may already be a basic knowledge of the operation. For outsiders or those unfamiliar with the operation, a brief description of the system and explanation of key issues should be provided. For an existing system, a facility walk-through is an excellent way of getting familiar with the operation.

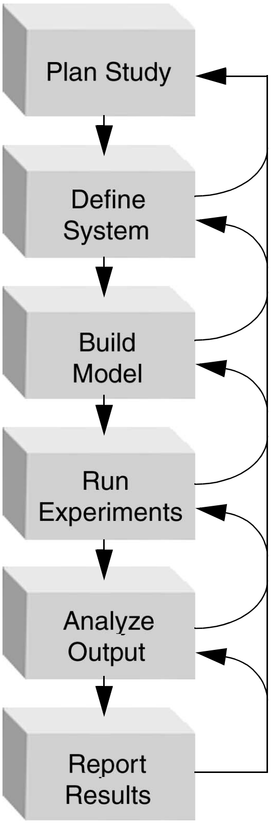
Once a suitable application or project has been identified as a candidate for simulation, decisions must be made about how to conduct the study. There are no strict rules on how to perform a simulation study, however, the following steps are generally recommended as a guideline (Shannon, 1975; Gordon, 1978; Law, 1991):

1. Plan the study
2. Define the system
3. Build the model

4. Run experiments
5. Analyze the output
6. Report results

Each step need not be completed in its entirety before moving on to the next step. The procedure for doing a simulation is an iterative one in which activities are refined and sometimes redefined with each iteration. Describing this iterative process, Pritsker and Pegden (1979) observe the following.

The iterative nature of this process is shown next:



Procedure for Conducting a Simulation Study

While the requirements for each step vary from simulation to simulation, the basic procedure is essentially the same. The primary value of adopting this systematic procedure, or one like it, is to ensure that the project is conducted in an organized, timely fashion with minimal waste of time and resources and maximum effectiveness in achieving the objectives.

Step 1: Planning the Study

Many simulation projects are doomed to failure from the outset due to poor planning. Undefined objectives, unrealistic expectations and a general lack of understanding of requirements frequently result in frustration and disappointment. If a simulation project is to be successful, a plan must be developed which is realistic, clearly communicated and closely followed. Planning a simulation study involves the following sub tasks:

- Defining Objectives
- Identifying Constraints
- Preparing a Simulation Specification
- Developing a Budget and Schedule

Each of these tasks is discussed in the following.

Defining Objectives

With a basic understanding of the system operation and an awareness of the issues of concern or interest, one or more objectives can be defined for the study. Simulation should only be used if an objective can be clearly stated and it is determined that simulation is the most suitable tool for achieving the objective. Defining an objective does not necessarily mean that there needs to be a problem to solve. A perfectly valid objective may be to see if there are, in fact, any unforeseen problems. Common types of objectives for a simulation study include the following:

- **Performance Analysis** How well does the system perform under a given set of circumstances in all measures of significance (utilization, throughput, waiting times, etc.)?
- **Capacity Analysis** What is the maximum processing or production capacity of the system?
- **Capability Analysis** Is the system capable of meeting specific performance requirements (throughput, waiting times, etc.), and, if not, what changes (added resources or improved methods) are recommended for making it capable?
- **Comparison Study** How well does one system configuration or design variation perform compared to another?
- **Sensitivity Analysis** Which decision variables are the most influential on one or more performance measures, and how influential are they?
- **Optimization Study** What combination of feasible values for a given set of decision variables best achieves desired performance objectives?
- **Decision/Response Analysis** What are the relationships between the values of one or more decision variables and the system response to those changes?
- **Constraint Analysis** Where are the constraints or bottlenecks in the system and what are workable solutions for either reducing or eliminating those constraints?
- **Communication Effectiveness** What variables and graphic representations can be used to most effectively depict the dynamic behavior or operation of the system?

Defining the objective should take into account what the ultimate intended use of the model will be. Some models are built as “throw-away” models to be used only once and then discarded. Other models are built to be used on an ongoing basis for continued “what-if” analyses. Some models need only provide a quantitative answer. Others require realistic animation to convince a skeptical customer. Some models are intended for use by only the analyst. Other models are intended for use by managers with little simulation background and must be easy to use. Some models are used to make decisions of minor consequence. Other models are relied upon to make major financial decisions.

Realizing the objectives of a simulation should consider both the purpose as well as the intended use of the model, the following questions should be asked when defining the objectives of the study:

- Why is the simulation being performed?
- Who will be using the model?
- To whom will the results of the simulation be presented?
- What information is expected from the model?
- Is this a “throw-away” model?
- How important is the decision being made?

Identifying Constraints

Equally as important as defining objectives is identifying the constraints under which the study must be conducted. It does little good if simulation solves a problem if the time to do the simulation extends beyond the deadline for applying the solution, or if the cost to find the solution exceeds the benefit derived. Objectives need to be tempered by the constraints under which the project must be performed such as the budget, deadlines, resource availability, etc. It is not uncommon to begin a simulation project with aspirations of developing an impressively detailed model or of creating a stunningly realistic animation only to scramble at the last minute, throwing together a crude model that barely meets the deadline.

Constraints should not always be viewed as an impediment. If no deadlines or other constraints are established, there is a danger of getting too involved and detailed in the simulation study and run the risk of "paralysis from analysis." The scope of any project has a tendency to shrink or expand to fill the time allotted.

In identifying constraints, anything that could have a limiting effect on achieving the desired objectives should be considered. Specific questions to ask when identifying constraints for a simulation study include the following:

- What is the budget for doing the study?
- What is the deadline for making the decision?
- What are the skills of those doing the study?
- How accessible is the input data?
- What computer(s) will be used for the study?

Preparing a Simulation Specification

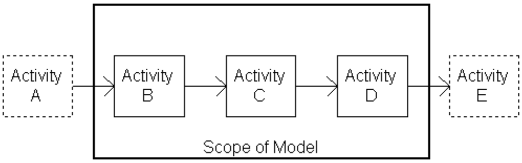
With clearly defined objectives and constraints, the simulation requirements can be specified. Defining a specification for the simulation is essential to projecting the time and cost needed to complete the study. It also guides the study and helps set expectations by clarifying to others exactly what the simulation will include or exclude. A specification is especially important if the simulation is being performed by an outside consultant so that you will know exactly what you are getting for your money. Aspects of the simulation project to be contained in the specification include the following:

- Scope
- Level of detail
- Degree of accuracy
- Type of experimentation
- Form of results

Each of these specification criteria will be discussed in the following pages.

Scope The scope refers to the breadth of the model or how much of the system the model will encompass. Determining the scope of the model should be based on how much bearing or impact a particular activity has on achieving the objectives of the simulation. A common tendency is to model the entire system, even when the problem area and all relevant variables are actually isolated within a smaller subsystem. If, for example, the objective is to find the number of operators required to meet a required production level for a machining cell, it is probably not necessary to model what happens to parts after leaving the cell.

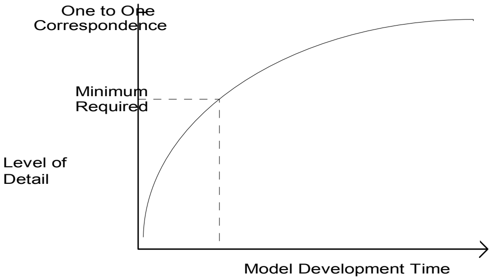
The following figure illustrates how the scope of the model should be confined to only those activities whose interactions have a direct bearing on the problem being studied. Upstream and downstream activities that do not impact the performance measure of interest should be omitted from the model. In the following figure, since the output rate from activity A is predictable, it can be modeled as simply an arrival rate to activity B. Since activity E never constrains output from activity D, it can also be ignored.



Confining the Scope to Impacting Activities

Level of Detail Project the level of detail defines the depth or resolution of the model. At one extreme, an entire factory can be modeled as a single "black box" operation with a random activity time. At the other extreme, every detailed motion of a machine could be modeled with a one-to-one correspondence depicting the entire machine operation.

Unlike the model scope which affects only the size of the model, the level of detail affects model complexity as well as model size. Determining the appropriate level of detail is an important decision. Too much detail makes it difficult and time consuming to develop a valid model. Too little detail may make the model too unrealistic by excluding critical variables. The following figure illustrates how the time to develop a model is affected by the level of detail. It also highlights the importance of including only enough detail to meet the objectives of the study.



Effect of Level of Detail on Model Development Time

The level of detail is determined largely by the degree of precision required in the output. If only a rough estimate is being sought, it may be sufficient to model each activity by its time, rather than specific details making up the activity. If, on the other hand, details such as downtimes or move times have an appreciable effect on the outcome of the model, they should be included.

Degree of Accuracy The degree of accuracy pertains to the correctness of the data being used. For some models or activities, the information need not be as accurate or exact as it does for others. The required degree of accuracy is determined by the objectives of the study. If the decision is important or a comparison is close, greater accuracy may be required. Accuracy sometimes has to be sacrificed if reliable information is simply unavailable such as when modeling a completely new system.

The required degree of accuracy can have a significant impact on the time and effort required to gather data. It often has little impact, however, on the model building time since a model can be built with estimated values that can later be replaced with more accurate values. Output precision is often governed by the degree of accuracy of the model.

Type of Experimentation The number and nature of the alternative solutions to be evaluated should be planned from the outset in order to ensure that adequate time is allotted. This decision is often influenced by the deadline constraints of the study. Where alternatives with only slight differences are to be evaluated, a base model can be developed requiring only minor modifications to model each alternative. If alternative configurations are significantly different, it may require nearly as much effort modeling each configuration as it does developing the initial model.

For studies in which improvements to an existing system are being considered, it is often helpful and effective to model the current system as well as the proposed system. The basic premise is that you are not ready to make improvements to a system until you understand how the current system operates. Information on the current system is easier to obtain than information on areas of change. Once a model of the current system is built, it is often easier to visualize what changes need to be made for the modified system. Both systems may even be modeled together in the same simulation and made to run side by side. During the final presentation of the results, being able to show both “as is” and “to be” versions of the system effectively demonstrates the impact changes can have on system performance.

Form of Results The form in which the results are to be presented can significantly affect the time and effort involved in the simulation study. If detailed animation or an extensive report is expected, the project can easily stretch on for several weeks after the experimental phase has been completed. Many times the only result required is a simple verification of whether a system is capable of meeting a production requirement. In such cases, a simple answer will suffice.

Developing a Budget and Schedule

With objectives and constraints clearly defined and a specification prepared identifying the work to be performed, a budget and schedule should be developed projecting the expected cost and time for completing the simulation project. Obviously, the time to perform a simulation project will vary depending on the size and difficulty of the project. If data is not readily available, it may be necessary to add several additional weeks to the project. A small project can take two to four weeks while large projects can take two to four months. A simulation schedule should be based on realistic projections of the time requirements keeping in mind the following:

- Defining the system to be modeled can take up to 50% of the project time.
- Model building usually takes the least amount of time (10 to 20%).
- Once a base model is built, it can take several weeks to conduct all of the desired experiments, especially if alternative designs are being compared.

While it may have initially been determined that simulation is a suitable tool for achieving the objective, the decision to use simulation may need to be reconsidered in light of projected cost and time estimates. Simulation may be a good solution to the problem at hand, but if the time or the cost of doing the project outweighs the anticipated benefits, either an alternative solution may need to be explored or the objectives may need to be modified to cut down on the level of effort required.

Step 2: Defining the System

With clearly defined objectives and a well organized plan for the study, the system that will be simulated can begin to be defined in detail. This can be viewed as the development of a conceptual model on which the simulation model will be based. The process of gathering and validating system information can be overwhelming when faced with the stacks of uncorrelated data to sort through. Data is seldom available in a form that defines exactly how the system works. Many data gathering efforts end up with lots of data but very little useful information.

Data gathering should never be performed without a purpose. Rather than being haphazard, data gathering should be goal oriented with a focus on information that will achieve the objectives of the study. There are several guidelines to keep in mind when gathering data.

- **Identify cause-and-effect relationships** It is important to correctly identify the causes or conditions under which activities are performed. In gathering downtime data, for example, it is helpful to distinguish between downtimes due to equipment failure or personal emergencies and planned downtimes for break. Once the causes have been established and analyzed, activities can be properly categorized.

- **Look for key impact factors** Discrimination should be used when gathering data to avoid wasting time examining factors that have little or no impact on system performance. If, for example, an operator is dedicated to a particular task and, therefore, is never a cause of delays in service, there is no need to include the operator in the model. Likewise, extremely rare downtimes, negligible move times and other insignificant or irrelevant activities that have no appreciable effect on routine system performance may be safely ignored.

- **Distinguish between time and condition dependent activities** Time-dependent activities are those that take a predictable amount of time to complete, such as customer service. Condition-dependent activities can only be completed when certain defined conditions within the system are satisfied. Because condition-dependent activities are uncontrollable, they are unpredictable. An example of a condition-dependent activity might be the approval of a loan application contingent upon a favorable credit check.

  Many activities are partially time-dependent and partially condition-dependent. When gathering data on these activities, it is important to distinguish between the time actually required to perform the activity and the time spent waiting for resources to become available or other conditions to be met before the activity can be performed. If, for example, historical data is used to determine repair times, the time spent doing the actual repair work should be used without including the time spent waiting for a repair person to become available.

- **Focus on essence rather than substance** A system definition for modeling purposes should capture the key cause-and-effect relationships and ignore incidental details. Using this "black box" approach to system definition, we are not concerned about the nature of the activity being performed, but only the impact that the activity has on the use of resources and the delay of entity flow. For example, the actual operation performed on a machine is not important, but only how long the operation takes and what resources, if any, are tied up during the operation. It is important for the modeler to be constantly thinking abstractly about the system operation in order to avoid getting too caught up in the incidental details.

- **Separate input variables from response variables** Input variables in a model define how the system works (e.g., activity times, routing sequences, etc.). Response variables describe how the system responds to a given set of input variables (e.g., work-in-process, idle times, resource utilization, etc.). Input variables should be the focus of data gathering since they are used to define the model. Response variables, on the other hand, are the output of a simulation. Consequently, response variables should only be gathered later to help validate the model once it is built and run.

These guidelines should help ensure that the system is thought of in the proper light for simulation purposes.

To help organize the process of gathering data for defining the system, the following steps are recommended:

- Determine data requirements.
- Use appropriate data sources.
- Make assumptions where necessary.
- Convert data into a useful form.
- Document and approve the data.

Each of these steps is explained on the following pages.

Determining Data Requirements

The first step in gathering system data is to determine what data is required for building the model. This should be dictated primarily by the scope and level of detail required to achieve the model objectives as described earlier. It is best to go from general to specific in gathering system data. The initial focus should be on defining the overall process flow to provide a skeletal framework for attaching more detailed information. Detailed information can then be added gradually as it becomes available (e.g., resource requirements, processing times, etc.). Starting with the overall process flow not only provides an orderly approach to data gathering, but also enables the model building process to get started which reduces the amount of time to build and debug the model later. Often, missing data becomes more apparent as the model is being built.

In defining the basic flow of entities through the system, a flow diagram can be useful as a way of documenting and visualizing the physical flow of entities from location to location. Once a flow diagram is made, a structured walk-through can be conducted with those familiar with the operation to ensure that the flow is correct and that nothing has been overlooked. The next step might be to define the detail of how entities move between locations and what resources are used for performing operations at each location. At this point it is appropriate to identify location capacities, move times, processing times, etc.

To direct data gathering efforts and ensure that meetings with others, on whom you depend for model information, are productive, it may be useful to prepare a specific list of questions that identify the data needed. A list of pertinent questions to be answered might include the following:

1. What types of entities are processed in the system and what attributes, if any, distinguish the way in which entities of the same type are processed or routed?
2. What are the route locations in the system (include all places where processing or queuing occurs, or where routing decisions are made) and what are their capacities (i.e., how many entities can each location accommodate or hold at one time)?
3. Besides route locations, what types of resources (personnel, vehicles, equipment) are used in the system and how many units are there of each type (resources used interchangeably may be considered the same type)?
4. What is the routing sequence for each entity type in the system?
5. What activity, if any, takes place for each entity at each route location (define in terms of time required, resources used, number of entities involved and any other decision logic that takes place)?
6. Where, when and in what quantities do entities enter the system (define the schedule, interarrival time, cyclic arrival pattern, or condition which initiates each arrival)?
7. In what order do multiple entities depart from each location (First in, First out; Last in, First out)?
8. In situations where an output entity could be routed to one of several alternative locations, how is the routing decision made (e.g., most available capacity, first available location, probabilistic selection)?
9. How do entities move from one location to the next (define in terms of time and resources required)?
10. What triggers the movement of entities from one location to another (i.e., available capacity at the next location, a request from the downstream location, an external condition)?
11. How do resources move from location to location to perform tasks (define either in terms of speed and distance, or time)?

12. What do resources do when they finish performing a task and there are no other tasks waiting (e.g., stay put, move somewhere else)?
13. In situations where multiple entities could be waiting for the same location or resource when it becomes available, what method is used for making an entity selection (e.g., longest waiting entity, closest entity, highest priority, preemption)?
14. What is the schedule of availability for resources and locations (define in terms of shift and break schedules)?
15. What planned interruptions do resources and locations have (scheduled maintenance, setup, changeover)?
16. What random failures do resources and locations experience (define in terms of distributions describing time to failure and time to repair)?

Depending on the purpose of the simulation and level of detail needed, some of these questions may not be applicable. For very detailed models additional questions may need to be asked. Answers to these questions should provide nearly all of the information necessary to build a model.

Using Appropriate Data Sources

Having a specific set of questions for defining the system, we are now ready to search for the answers. Information seldom comes from a single source. It is usually the result of reviewing reports, conducting personal interviews, personal observation and making lots of assumptions. “It has been my experience,” notes Carson (1986), “that for large-scale real systems, there is seldom any one individual who understands how the system works in sufficient detail to build an accurate simulation model. The modeler must be willing to be a bit of a detective to ferret out the necessary knowledge.” Good sources of system data includes the following:

- Time Studies
- Predetermined Time Standards
- Flow Charts
- Facility Layouts
- Market Forecasts
- Maintenance Reports
- On-line tracking systems
- Equipment Manufacturers
- Managers
- Engineers
- Facility Walk-throughs
- Comparisons with Similar Operations

In deciding whether to use a particular source of data, it is important to consider the relevancy, reliability and accessibility of the source. If the information that a particular source can provide is irrelevant for the model being defined, then that source should not be consulted. What good is a maintenance report if it has already been decided that downtimes are not going to be included in the model? Reliability of the source will determine the validity of the model. A managers perception, for example, may not be as reliable as actual production logs. Finally, if the source is difficult to access, such as a visit to a similar facility in a remote site, it may have to be omitted.

Making Assumptions

Not long after data gathering has started, you may realize certain information is unavailable or perhaps unreliable. Complete, accurate, and up-to-date data for all the information needed is rarely obtainable, especially when modeling a new system about which very little is known. For system elements about which little is known, assumptions must be made. There is nothing wrong with assumptions as long as they can be agreed upon, and it is recognized that they are *only* assumptions. Any design effort must utilize assumptions where complete or accurate information is lacking.

Many assumptions are only temporary until correct information can be obtained or it is determined that more accurate information is necessary. Often, *sensitivity analysis*, in which a range of values is tested for potential impact, can give an indication of just how accurate the data really needs to be. A decision can then be made to firm up the assumptions or to leave them alone. If, for example, the degree of variation in a particular activity time has little or no impact on system performance, then a constant activity time may be used. Otherwise, it may be important to define the exact distribution for the activity time.

Another approach in dealing with assumptions is to run three different scenarios showing a "best-case" using the most optimistic value, a "worst-case" using the most pessimistic value, and a "most-likely-case" using a best-estimate value. This will help determine the amount of risk you want to take in assuming a particular value.

Converting Data to a Useful Form

Data is seldom in a form ready for use in a simulation model. Usually, some analysis and conversion needs to be performed for data to be useful as an input parameter to the simulation. Random phenomena must be fitted to some standard, theoretical distribution such as a normal or exponential distribution (Law and Kelton, 1991), or be input as a frequency distribution. Activities may need to be grouped together to simplify the description of the system operation.

Distribution Fitting To define a distribution using a theoretical distribution requires that the data, if available, be fit to an appropriate distribution that best describes the variable. ProModel includes the Stat::Fit distribution fitting package which assists in fitting sample data to a suitable theoretical distribution. An alternative to using a standard theoretical distribution is to summarize the data in the form of a frequency distribution that can be used directly in the model. A frequency distribution is sometimes referred to as an *empirical* or *user-defined* distribution.

Whether fitting data to a theoretical distribution, or using an empirical distribution, it is often useful to organize the data into a frequency distribution table. Defining a frequency distribution is done by grouping the data into intervals and stating the frequency of occurrence for each particular interval. To illustrate how this is done, the following frequency table tabulates the number and frequency of observations for a particular activity requiring a certain range of time to perform.

Frequency Distributions of Delivery Times

Delivery Time (days)	Number of Observations	Percentage	Cumulative Percentage
0 - 1	25	16.5	16.5
1 - 2	33	21.7	38.2
2 - 3	30	19.7	57.9
3 - 4	22	14.5	72.4
4 - 5	14	9.2	81.6
5 - 6	10	6.6	88.2
6 - 7	7	4.6	92.8
7 - 8	5	3.3	96.1
8 - 9	4	2.6	98.7
9 - 10	2	1.3	100.0

Total Number of Observations = 152

While there are rules that have been proposed for determining the interval or cell size, the best approach is to make sure that enough cells are defined to show a gradual transition in values, yet not so many cells that groupings become obscured.

Note in the last column of the frequency table that the percentage for each interval may be expressed optionally as a cumulative percentage. This helps verify that all 100% of the possibilities are included.

When gathering samples from a static population, one can apply descriptive statistics and draw reasonable inferences about the population. When gathering data from a dynamic and possibly time varying system, however, one must be sensitive to trends, patterns, and cycles that may occur with time. The samples drawn may not actually be homogenous samples and, therefore, unsuitable for applying simple descriptive techniques.

Activity Grouping Another consideration in converting data to a useful form is the way in which activities are grouped for modeling purposes. Often it is helpful to group activities together so long as important detail is not sacrificed. This makes models easier to define and more manageable to analyze. In grouping multiple activities into a single activity time for simplification, consideration needs to be given as to whether activities are performed in parallel or in series. If activities are done in parallel or with any overlap, the time during which overlapping occurs should not be additive.

Serial activities are always additive. For example, if a series of activities is performed on an entity at a location, rather than specifying the time for each activity, it may be possible to sum activity times and enter a single time or time distribution.

Documenting and Approving the Data

When it is felt that all relevant information has been gathered and organized into a usable form, it is advisable to document the information in the form of data tables, relational diagrams and assumption lists. Sources of data should also be noted. This document should then be reviewed by others who are in a position to evaluate the validity of the data and approve the assumptions made. This document will be helpful later if you need to make modifications to the model or look at why the actual system ends up working differently than what was modeled.

In addition to including those factors to be used in the model, the data document should also include those factors deliberately excluded from the model because they are deemed to be either insignificant or irrelevant. If, for example, break times are not identified in the system description, a statement of explanation should be made explaining why. Stating why certain factors are being excluded from the system description will help resolve later concerns that may question why the factors were omitted.

Validating system data can be a time-consuming and difficult task, especially when many assumptions are made. In practice, data validation ends up being more of a consensus or agreement that is obtained confirming that the information is good enough for the purposes of the model. While this approved data document provides the basis for building the model, it often changes as model building and experimentation get under way.

Step 3: Building the Model

Once sufficient information has been compiled to define the basic system operation, the model building activity can begin. While starting to build a model too early can be a wasted exercise, waiting until all of the information is completely gathered and validated may unnecessarily postpone the building of the model. Getting the model started before the data is completely gathered may even help identify missing information needed to proceed.

The goal of model building is to provide a valid representation of the defined system operation.

Additionally, the model must be able to provide any other statistical or graphical representation needed to satisfy the objectives of the study. A model is neither true nor false, but rather useful or not useful. Once validated, a model is useful when it provides the needed information to meet the objectives of the simulation.

Progressive Refinement

One nice feature of simulation is that models do not have to include all of the final detail before they will run. This allows a progressive refinement strategy to be used in which detail is added to the model in stages rather than all at once. Not only do models get built and running quicker this way, but it also makes models easier to debug. In the initial stages of a model, for example, attractive graphics are not very useful and, since they are likely to be changed anyway, should not be added until later when preparing for the final model presentation.

The complexity of model building should never be underestimated and it is always better to begin simple and add complexity rather than create an entire complex model at once. It is also easier to add detail to a model than it is to remove it from a model. Building a model in stages enables bugs to be more readily identified and corrected. Emphasizing the importance of applying progressive refinement to model building, Law and Kelton (1991) have advised:

Although there are few firm rules on how one should go about the modeling process, one point on which most authors agree is that it is always a good idea to start with a simple model which can later be made more sophisticated if necessary. A model should contain only enough detail to capture the essence of the system for the purposes for which the model is intended: it is not necessary to have a one-to-one correspondence between elements of the model and elements of the system. A model with excessive detail may be too expensive to program and to execute.

Incremental Expansion

In addition to adding complexity to a model in stages, models that have a broad scope are sometimes easier to build in phases where additional sections are added incrementally to the model. This method of "eating the elephant one bite at a time" allows a portion of the model to be built, tested and debugged before adding new sections and makes a large task more manageable.

For unusually large models, it may be useful to identify definable boundaries within a model to permit *model partitioning*. Model partitioning is the process of subdividing a model into two or more modules that represent physically separate sections within the system. The purpose of model partitioning is to allow model sections to be built and debugged, possibly even by separate individuals, independently of each other. Once sections are finished, they can be merged together to create the overall model. This “divide-and-conquer” method of model building can greatly reduce the time and difficulty in building and debugging large models.

Model Verification

Once a model is defined using a selected software tool, the model must generally be debugged to ensure that it works correctly. The process of demonstrating that a model works as intended is referred to in simulation literature as model *verification*. It is much easier to debug a model built in stages and with minimal detail than to debug a large and complex model. Eliminating bugs in a program model can take a considerable amount of time, especially if a general purpose programming language (e.g., C++) in which frequent coding errors occur is used. Most simulation languages provide a trace capability in the form of audit trails, screen messages, graphic animation, or a combination of all three. A trace enables the user to look inside of the simulation to see if the simulation is performing the way it should. Good simulation products provide interactive debugging capability which further facilitates the debugging process. A thorough "walk-through" of the model input is always advisable.

Model Validation

During the process of model building, the modeler must be constantly concerned with how closely the model reflects the system definition. The process of determining the degree to which the model corresponds to the real system, or at least accurately represents the model specification document, is referred to as model *validation*. Proving absolute validity is a non attainable goal. As Neelamkavil (1987) explains, "True validation is a philosophical impossibility and all we can do is either invalidate or fail to invalidate." For this reason, what we actually seek to establish is a high degree of*face validity.* Face validity means that, from all outward indications, the model appears to be an accurate representation of the system. From this standpoint, validating a model is the process of substantiating that the model, within its domain of applicability, is sufficiently accurate for the intended application (Schlesinger, 1979).

There is no simple test to establish the validity of a model. Validation is an inductive process through which the modeler draws conclusions about the accuracy of the model based on the evidence available. Gathering evidence to determine model validity is largely accomplished by examining the model structure (i.e., the algorithms and relationships) to see how closely it corresponds to the actual system definition. For models having complex control logic, graphic animation can be used effectively as a validation tool. Finally, the output results should be analyzed to see if the results appear reasonable. If circumstances permit, the model may even be compared to that actual system to see how they correspond. If these procedures are performed without encountering a discrepancy between the real system and the model, the model is said to have face validity.

Step 4: Conducting Experiments

The fourth step in a simulation study is to conduct simulation experiments with the model. Simulation is basically an application of the scientific method. In simulation, one begins with a theory of why certain design rules or management strategies are better than others. Based on these theories the designer develops a hypothesis which he tests through simulation. Based on the results of the simulation the designer draws conclusions about the validity of his hypothesis. In a simulation experiment there are input variables defining the model which are independent and may be manipulated or varied. The effects of this manipulation on other dependent or response variables are measured and correlated.

In some simulation experiments we are interested in the *steady-state* behavior of the model. Steady-state behavior does not mean that the simulation produces a steady outcome, but rather the distribution or statistical variation in outcome does not change over time. For example, a distribution warehouse may ship between 200 and 220 parts per hour under normal operating conditions. For many simulations we may only be interested in a particular time period, such as a specific day of the week. For these studies, the simulation may never reach a steady state.

As with any experiment involving a system with random characteristics, the results of the simulation will also be random in nature. The results of a single simulation run represent only one of several possible outcomes. This requires that multiple *replications* be run to test the reproducibility of the results. Otherwise, a decision might be made based on a fluke outcome, or at least an outcome not representative of what would normally be expected. Since simulation utilizes a pseudo-random number generator for generating random numbers, running the simulation multiple times simply reproduces the same sample. In order to get an independent sample, the starting seed value for each random stream must be different for each replication, ensuring that the random numbers generated from replication to replication are independent.

Depending on the degree of precision required in the output, it may be desirable to determine a *confidence interval* for the output. A confidence interval is a range within which we can have a certain level of confidence that the true mean falls. For a given confidence level or probability, say .90 or 90%, a confidence interval for the average utilization of a resource might be determined to be between 75.5 and 80.8%. We would then be able to say that there is a .90 probability that the true mean utilization of the modeled resource (not of the actual resource) lies between 75.5 and 80.8%.

Fortunately, ProModel provides convenient facilities for conducting experiments, running multiple replications and automatically calculating confidence intervals. The modeler must still decide, however, what types of experimentation are appropriate. When conducting simulation experiments, the following questions should be asked:

- Am I interested in the steady state behavior of the system or a specific period of operation?
- How can I eliminate start-up bias or getting the right initial condition for the model?
- What is the best method for obtaining sample observations that may be used to estimate the true expected behavior of the model?
- What is an appropriate run length for the simulation?
- How many replications should be made?
- How many different random streams should be used?
- How should initial seed values be controlled from replication to replication?

Answers to these questions will be determined largely by the following three factors:

1. The nature of the simulation (terminating or nonterminating).
2. The objective of the simulation (capacity analysis, alternative comparisons, etc.).
3. The precision required (rough estimate versus confidence interval estimates).

Terminating Versus Non-terminating Simulations

As part of setting up the simulation experiment, one must decide what type of simulation to run. Simulations are usually distinguished as being one of two types: *terminating* or *non-terminating*. The difference between the two has to do with whether we are interested in the behavior of the system over a particular period of time or in the steady-state behavior of the system. It has nothing to do, necessarily, with whether the system itself terminates or is ongoing. The decision to perform a terminating or non-terminating simulation has less to do with the nature of the system than it does with the behavior of interest.

A terminating simulation is one in which the simulation starts at a defined state or time and ends when it reaches some other defined state or time. An initial state might be the number of parts in the system at the beginning of a work day. A terminating state or event might be when a particular number of jobs have been completed. Consider, for example, an aerospace manufacturer that receives an order to manufacture 200 airplanes of a particular model. The company might be interested in knowing how long it will take to produce the aircraft along with existing workloads. The simulation run starts with the system empty and is terminated when the 200th plane is completed since that covers the period of interest. A point in time which would bring a terminating simulation to an end might be the closing of shop at the end of a business day, or the completion of a weekly or monthly production period. It may be known, for example, that a production schedule for a particular item changes weekly. At the end of each 40 hour cycle, the system is “emptied” and a new production cycle begins. In this situation, a terminating simulation would be run in which the simulation run length would be 40 hours.

Terminating simulations are not intended to measure the steady-state behavior of a system. In a terminating simulation, average measures are of little meaning. Since a terminating simulation always contains transient periods that are part of the analysis, utilization figures have the most meaning if reported for successive time intervals during the simulation.

A non-terminating or steady-state simulation is one in which the steady-state behavior of the system is being analyzed. A non-terminating simulation does not mean that the simulation never ends, nor does it mean that the system being simulated has no eventual termination. It only means that the simulation could theoretically go on indefinitely with no statistical change in behavior. For non-terminating simulations, the modeler must determine a suitable length of time to run the model. An example of a non-terminating simulation is a model of a manufacturing operation in which oil filters are produced on a continual basis at the same pace. The operation runs two shifts with an hour break during each shift in which everything momentarily stops. Break and third shift times are excluded from the model since work always continues exactly as it left off before the break or end of shift. The length of the simulation is determined by how long it takes to get a representative steady-state reading of the model behavior.

Running Terminating Simulations

Experiments involving terminating simulations are usually conducted by making several simulation runs, or replications, of the period of interest using a different random seed for each run. This procedure enables statistically independent and unbiased observations to be made on the system response over the period simulated. Statistics are often gathered on performance measures for successive intervals of time during the period.

For terminating simulations, we are usually interested in final production counts and changing patterns of behavior over time rather than the overall average behavior. It would be absurd, for example, to conclude that because two technicians are busy only an average of 40% during the day that only one technician is needed. This average measure reveals nothing about the utilization of the technicians during peak periods of the day. A more detailed report of waiting times during the entire work day may reveal that three technicians are needed to handle peak periods, whereas only one technician is necessary during off-peak hours. In this regard, Hoover and Perry (1990) note:

It is often suggested in the simulation literature that an overall performance be accumulated over the course of each replication of the simulation, ignoring the behavior of the systems at intermediate points in the simulation. We believe this is too simple an approach to collecting statistics when simulating a terminating system. It reminds us of the statistician who had his head in the refrigerator and feet in the oven, commenting that on the average he was quite comfortable.

For terminating simulations, the three important questions to answer in running the experiment are:

1. What should be the initial state of the model?
2. What is the terminating event or time?
3. How many replications will you make?

Many systems operate on a daily cycle, or, if a pattern occurs over a weeks time, the cycle is weekly. Some cycles may vary monthly or even annually. Cycles need not be repeating to be considered a cycle. Airlines, for example, may be interested in the start-up period of production during the introduction of a new airport which is a one-time occurrence.

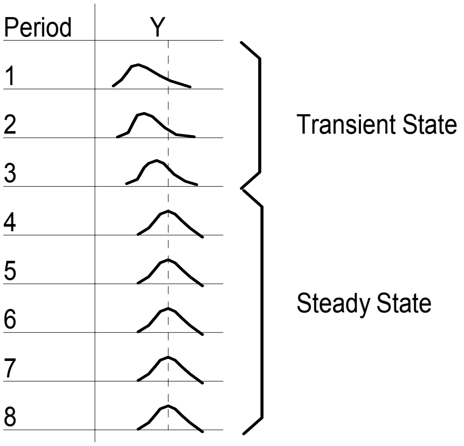
The number of replications should be determined by the precision required for the output. If only a rough estimate of performance is being sought, three to five replications are sufficient. For greater precision, more replications should be made until a confidence interval with which you feel comfortable is achieved.

Running Non-terminating Simulations

The issues associated with generating meaningful output statistics for terminating simulations are somewhat different that those associated with generating statistics for non-terminating systems. In steady-state simulations, we must deal with the following issues:

1. Determining the initial warm-up period.
2. Selecting among several alternative ways for obtaining sample observations.
3. Determining run length.

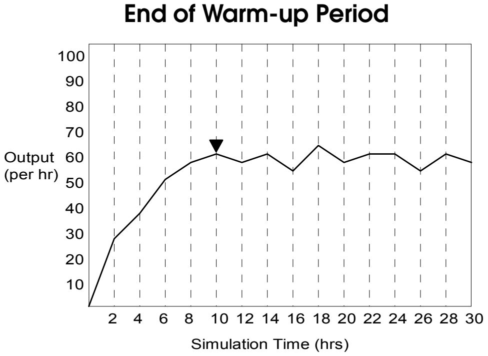
Determining the Warm-up Period In a steady-state simulation, we are interested in the steady-state behavior of the model. Since a model starts out empty, it usually takes some time before it reaches steady-state. In a steady-state condition, the response variables in the system (e.g., processing rates, utilization, etc.) exhibit statistical regularity (i.e., the distribution of these variables are approximately the same from one time period to the next). The following figure illustrates the typical behavior of a response variable, Y, as the simulation progresses through N periods of a simulation.



Behavior of Response Variable Y for Successive Periods During Simulation

The time that it takes to reach steady-state is a function of the activity times and the amount of activity taking place. For some models, steady-state might be reached in a matter of a few hours of simulation time. For other models it may take several hundred hours to reach steady-state. In modeling steady-state behavior we have the problem of determining when a model reaches steady-state. This start-up period is usually referred to as the *warm-up period*. We want to wait until after the warm-up period before we start gathering any statistics. This way we eliminate any bias due to observations taken during the transient state of the model.

While several methods have been presented for determining warm-up time (Law and Kelton, 1991), the easiest and most straightforward approach, although not necessarily the most reliable, is to run a preliminary simulation of the system, preferably with several (3 to 5) replications, and observe at what time the system reaches statistical stability. The length of each replication should be relatively long and allow even rarely occurring events, such as infrequent downtimes, to occur at least two or three times. To determine a satisfactory warm-up period using this method, one or more key response variables should be monitored by period over time, like the average number of entities in a queue or the average utilization of a resource. This approach assumes that the *mean* value of the monitored response variable is the primary indicator of convergence rather than the *variance*, which often appears to be the case. If possible, it is preferable to reset the response variable after each period rather than track the cumulative value of the variable, since cumulative plots tend to average out instability in data. Once these variables begin to exhibit steady-state, we can add a 20% to 30% safety factor and be reasonably safe in using that period as the warm-up period. This approach is simple, conservative and usually satisfactory. Remember, the danger is in underestimating the warm-up period, not overestimating it. Relatively little time and expense is needed to run the warm-up period longer than actually required. The following figure illustrates the average number of entities processed each hour for several replications. Since statistical stability is reached at about 10 hours, 12 to 15 hours is probably a safe warm-up period to use for the simulation.



Plot of Hourly Entity Output to Identify Start of Steady-State

Obtaining Sample Observations In a terminating simulation, sample observations are made by simply running multiple replications. For steady-state simulations, we have several options for obtaining sample observations. Two widely used approaches are running multiple replications and interval batching. The method supported in ProModel is running multiple replications.

Running multiple replications for non-terminating simulations is very similar to running terminating simulations. The only difference is that (1) the initial warm-up period must be determined, and (2) an appropriate run length must be determined. Once the replications are made, confidence intervals can be computed as described earlier in this chapter. One advantage of running independent replications is that samples are independent. On the negative side, running through the warm-up phase for each replication slightly extends the length of time to perform the replications. Furthermore, there is a possibility that the length of the warm-up period is underestimated, causing biased results.

Interval batching (also referred to as the *batch means technique*) is a method in which a single, long run is made with statistics being reset at specified time intervals. This allows statistics to be gathered for each time interval with a mean calculated for each interval batch. Since each interval is correlated to both the previous and the next intervals (called serial correlation or autocorrelation), the batches are not completely independent. The way to gain greater independence is to use large batch sizes and to use the mean values for each batch. When using interval batching, confidence interval calculations can be performed. The number of batch intervals to create should be at least 5 to 10 and possibly more depending on the desired confidence interval.

Determining Run Length Determining run length for terminating simulations is quite simple since there is a natural event or time point that defines it for us. Determining the run length for a steady-state simulation is more difficult since the simulation can be run indefinitely. The benefit of this, however, is that we can produce good representative samples. Obviously, running extremely long simulations is impractical, so the issue is to determine an appropriate run length that ensures a sufficiently representative sample of the steady-state response of the system is taken.

The recommended length of the simulation run for a steady-state simulation is dependent upon (1) the interval between the least frequently occurring event and (2) the type of sampling method (replication or interval batching) used. If running independent replications, it is usually a good idea to run the simulation enough times to let every type of event (including rare ones) happen at least a few times if not several hundred. Remember, the longer the model is run, the more confident you can become that the results represent a steady-state behavior. If collecting batch mean observations, it is recommended that run times be as large as possible to include at least 1000 occurrences of each type of event (Thesen and Travis, 1992).

Comparing Alternative Systems

Simulations are often performed to compare two or more alternative designs. This comparison may be based on one or more decision variables such as buffer capacity, work schedule, resource availability, etc. Comparing alternative designs requires careful analysis to ensure that differences being observed are attributable to actual differences in performance and not to statistical variation. This is where running multiple replications may again be helpful. Suppose, for example, that method A for deploying resources yields a throughput of 100 entities for a given time period while method B results in 110 entities for the same time period. Is it valid to conclude that method B is better than method A, or might additional replications actually lead the opposite conclusion?

Evaluating alternative configurations or operating policies can sometimes be performed by comparing the average result of several replications. Where outcomes are close or where the decision requires greater precision, a method referred to as *hypothesis testing* should be used. In hypothesis testing, first a hypothesis is formulated (e.g., that methods A and B both result in the same throughput) and then a test is made to see whether the results of the simulation lead us to reject the hypothesis. The outcome of the simulation runs may cause us to reject the hypothesis that methods A and B both result in equal throughput capabilities and conclude that the throughput does indeed depend on which method is used.

Sometimes there may be insufficient evidence to reject the stated hypothesis and thus the analysis proves to be inconclusive. This failure to obtain sufficient evidence to reject the hypothesis may be due to the fact that there really is no difference in performance, or it may be a result of the variance in the observed outcomes being too high given the number of replications to be conclusive. At this point, either additional (perhaps time consuming) replications may be run or one of several variance reduction techniques might be employed (see Law and Kelton, 1991).

Factorial Design

In simulation experiments we are often interested in finding out how different input variable settings impact the response of the system. Rather than run hundreds of experiments for every possible variable setting, experimental design techniques can be used as a “short-cut” to finding those input variables of greatest significance. Using experimental-design terminology, input variables are referred to as *factors*, and the output measures are referred to as *responses*. Once the response of interest has been identified and the factors that are suspected of having an influence on this response defined, we can use a *factorial design* method which prescribes how many runs to make and what level or value to be used for each factor. As in all simulation experiments, it is still desirable to run multiple replications for each factor level and use confidence intervals to assess the statistical significance of the results.

One's natural inclination when experimenting with multiple factors is to test the impact that each individual factor has on system response. This is a simple and straightforward approach, but it gives the experimenter no knowledge of how factors interact with each other. It should be obvious that experimenting with two or more factors together can affect system response differently than experimenting with only one factor at a time and keeping all other factors the same.

One type of experiment that looks at the combined effect of multiple factors on system response is referred to as a *two-level, full-factorial design*. In this type of experiment, we simply define a high and low level setting for each factor and, since it is a full-factorial experiment, we try every combination of factor settings. This means that if there are five factors and we are testing two different levels for each factor, we would test each of the $2^5 = 32$ possible combinations of high and low factor levels. For factors that have no range of values from which a high and low can be chosen, the high and low levels are arbitrarily selected. For example, if one of the factors being investigated is an operating policy for doing work (e.g., first come, first served; or last come, last served), we arbitrarily select one of the alternative policies as the high level setting and a different one as the low level setting.

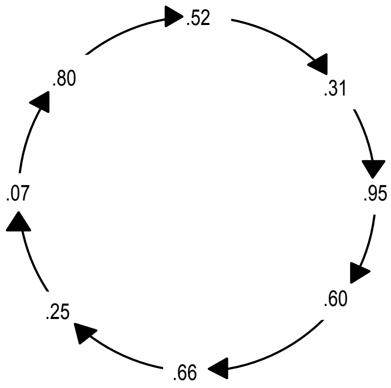
For experiments in which a large number of factors are being considered, a two-level full-factorial design would result in an extremely large number of combinations to test. In this type of situation, a *fractional-factorial design* is used to strategically select a subset of combinations to test in order to “screen out” factors with little or no impact on system performance. With the remaining reduced number of factors, more detailed experimentation such as a full-factorial experiment can be conducted in a more manageable fashion.

After fractional-factorial experiments and even two-level full-factorial experiments have been performed to identify the most significant factor level combinations, it is often desirable to conduct more detailed experiments, perhaps over the entire range of values, for those factors that have been identified as being the most significant. This provides more precise information for making decisions regarding the best factor values or variable settings for the system. For a more concise explanation of the use of factorial design in simulation experimentation see Law and Kelton (1991).

Use of Random Streams

One of the most valuable characteristics of simulation is the ability to reproduce and randomize replications of a particular model. Simulation allows probabilistic phenomena within a system to be controlled or randomized as desired for conducting controlled experiments. This control is made available through the use of *random streams*.

A stream is a sequence of independently cycling, unique random numbers uniformly distributed between 0 and 1 (see the figure on next page). Random number streams are used to generate additional random numbers from other probability distributions (Normal, Beta, Gamma). After sequencing through all of the random numbers in the cycle, the cycle starts over again with the same sequence. The length of the cycle before it repeats is called the *cycle period* and is usually very long.



Example of a Random Stream Cycle with a Very Short Period

A random stream is generated using a random number generator or equation. The random number generator begins with an initial seed value after which, each successive value uses the previous value as input to the generator. Each stream used in a simulation has its own independent seed and tracks its own values for subsequent input to the generator. Where the sequence begins in the cycle depends on the initial seed value used by the generator.

Any time a particular number seeds a stream, the same sequence of values will be repeated every time the same seed is used to initialize the stream. This means that various elements within a model can be held constant with respect to their performance while other elements vary freely. Simply specify one random number stream for one set of activities and another random number stream for all other activities.

Because the same seed produces the same sequence of values every time it is used, completely independent functions within a model must have their own streams from the start. For example, arrival distributions should generally have a random number stream used nowhere else in the entire model. That way, activities added to a model that sample from a random number stream will not inadvertently alter the arrival pattern because they do not affect the sample values generated from the arrival distribution.

To show an example of how multiple streams can be useful, consider two copy machines, Copy1 and Copy2, which go down approximately every 4 hours for servicing. To model this, the frequency or time between failures is defined by a normal distribution with a mean value of 240 minutes and a standard deviation of 15 minutes, N(240,15). The time to repair is 10 minutes. If no stream is specified in the normal distribution, the same stream will be used to generate sample values for both machines. So, if the next two numbers in the stream number are .21837 and .86469, Copy1 will get a sample value from the normal distribution that is different from Copy2. Therefore, the two machines will go down at different times.

Suppose, however, that the resource servicing the machines must service them both at the same time, so we would like to have the machines go down at the same time. Using the same stream to determine both downtimes will not bring them down at the same time, because a different random number will be returned from the stream with each call to generate a random normal variate. Using two different streams, each dedicated to a machine's downtime and each having the same initial seed, will ensure that both machines go down at the same time every time. The two streams have the same starting seed value so they will produce exactly the same sequence of random numbers.

Step 5: Analyzing the Output

Output analysis deals with drawing inferences about the actual system based on the simulation output. When conducting simulation experiments, extreme caution should be used when interpreting the simulation results. Since the results of a simulation experiment are random (given the probabilistic nature of the inputs), an accurate measurement of the statistical significance of the output is necessary.

People doing simulation in academia are often accused of working with contrived and often oversimplified assumptions, yet are extremely careful about ensuring the statistical significance of the model results. Simulation practitioners in industry, on the other hand, are usually careful to obtain valid model data, only to ignore the statistical issues associated with simulation output. Maintaining a proper balance between establishing model validation and establishing the statistical significance of simulation output is an important part of achieving useful results.

The most valuable benefit from simulation is to gain insight, not necessarily to find absolute answers. With this in mind, one should be careful about getting too pedantic about the precision of simulation output. With more than 60 combined years of experience in doing simulation modeling, Conway, Maxwell and Worona (1986) caution that attaching a statistical significance to simulation output can create a delusion that the output results are either more or less significant than they really are. They emphasize the practical, intuitive reading of simulation results. Their guideline is "If you can't see it with the naked eye, forget it."

The goal of conducting experiments is not just to find out how well a particular system operates, but hopefully to gain enough insight to be able to improve the system. Unfortunately, simulation output rarely identifies causes of problems, but only reports the symptomatic behavior of problems. Bottleneck activities, for example, are usually identified by looking for locations or queues that are nearly always full which feed into one or more locations that are sometimes empty. Detecting the *source* of the bottleneck is sometimes a bit trickier than identifying the bottleneck itself. Bottlenecks may be caused by excessive operation times, prolonged delays due to the unavailability of resources, or an inordinate amount of downtime. The ability to draw correct inferences from the results is essential to making system improvements.

Step 6: Reporting the Results

The last step in the simulation procedure is to make recommendations for improvement in the actual system based on the results of the simulated model. These recommendations should be supported and clearly presented so that an informed decision can be made. Documentation of the data used, the model(s) developed and the experiments performed should all be included as part of a final simulation report.

A simulation has failed if it produces evidence to support a particular change which is not implemented; especially if it is economically justified. The process of selling simulation results is largely a process of establishing the credibility of the model. It is not enough for the model to be valid, the client or management must also be *convinced* of its validity if it is to be used as an aid in decision making. Finally, the results must be presented in terms that are easy to understand and evaluate. Reducing the results to economic factors always produces a compelling case for making changes to a system.

In presenting results it is important to be sensitive to the way in which recommendations are made. It helps to find out whether recommendations are being sought or whether a simple summary of the results is wanted. It is generally wise to present alternative solutions and their implications for system performance without suggesting one alternative over another, particularly when personnel changes or cuts are involved. In fact, where there may be careers on the line, it is best to caution the decision maker that your simulation study looks only at the logistical aspects of the system and that it does not take into account the potential reactions or potential difficulties employees may have in accepting a particular solution.

Animation and output charts have become an extremely useful aid in communicating the results of a simulation study. This usually requires that some touch-up work be done to create the right effect in visualizing the model being simulated. In preparing the results, it is often necessary to add a few touch-ups to the model (like a full dress-rehearsal) so the presentation effectively and convincingly presents the results of the simulation study.

After the presentation is finished and there is no further analysis to be conducted (the final presentation always seems to elicit further suggestions for trying this or that with the model), the model recommendations, if approved, are ready to be implemented. If the simulation has been adequately documented, it should provide a good functional specification for the implementation team.

Pitfalls in Simulation

If the steps that have been outlined are followed, the chances of performing a successful simulation project are very good. Typical reasons why simulation projects fail include the following:

- Failure to state clear objectives at the outset.
- Failure to involve individuals affected by outcome.
- Overrunning budget and time constraints.
- Failure to document and get a consensus on input data.
- Including more detail than is needed.
- Including variables that have little or no impact on system behavior.
- Failure to verify and validate the model.
- Basing decisions on a single run observation.
- Basing decisions on average statistics when the output is actually cyclical.
- Being too technical and detailed in presenting the results to management.

Summary

A simulation project has distinct phases that must be understood and followed in order to be successful. Simulation requires careful planning with realistic goals and expectations. Steps to performing a simulation study include planning the study, defining the system, building the model, conducting experiments, analyzing the output, and presenting the results. Systematically following these steps will help avoid the pitfalls that frequently occur when conducting a simulation study.

Building a Model

ProModel gives you the flexibility to create a model in several ways—the easiest is to use the graphical point-and-click approach. To build a model, you should first define any locations in the system. With locations defined, you are ready to create entities (parts, customers, calls, etc.) and schedule their arrivals to the locations you created. Next, specify the process logic for entities at each location (this will establish the entity flow throughout the model). Finally, define any optional model elements such as attributes, variables, or arrays that you will use in processing entities.

Modeling Scenario

Before we actually begin building a model, let's look at a fictitious scenario for our model building session.

Cogswell Cogs has just secured a contract to produce a new cog for production of the X-95C Family Space Cruiser. The current capacity of the Cogswell facility is not adequate to handle any additional work load while continuing to fill existing orders. Therefore, Mr. Cogswell has ordered the I.E. department to simulate the design of a new workcell dedicated to the production of the new cog.

The process consists of loading a cast blank onto an NC mill for milling of the outside splines. Once the splines have been cut, the cog must be degreased, inspected and loaded with an inner bearing. All operations, including inspection, are to be performed by a single operator.

Model Elements

In building this model we must define all of the basic modeling elements and a few of the optional elements.

Locations

We need some type of receiving location to hold incoming entities. We also need processing locations where entities have value added to them. For the given production rate, Cogswell's engineers have determined that the workcell will require two NC_300 series numerically controlled mills, a degreasing machine, and an inspection/assembly station.

Entities (Parts)

The entity types in this system include Pallets, each carrying six cast Blanks. Blanks become Cogs after processing, and Bearings are loaded at the Inspect station. If a Cog fails the inspection it will be called a Reject.

Arrivals

Cogswell's engineers have determined that Pallets should arrive at the rate of 1 Pallet every 45 minutes.

Processing

The operation at each mill requires an operator to load the Blank, which takes a normally distributed amount of time with a mean of 3 minutes and standard deviation of .2 minutes (i.e., N(3,.2)). After a blank has been loaded, the machining time is a constant 5.7 minutes.

Cogs are then removed from the mill and placed in the degreasing machine. The degreasing machine has capacity for 2 Cogs, and has a cycle time of 5 minutes.

Once the Cogs have been degreased, they are inspected for proper spline depth, and a bearing is installed in the center hole. This process requires the cell operator, and takes U(3.2,.3) minutes for the inspection and U(1.5,.2) minutes for the Bearing to be installed. If the Cog fails inspection, no Bearing is installed.

Resources

A single operator, CellOp, performs all manual operations.

Path Networks

In order to make CellOp a mobile resource, we must define a path network. We'll call it CellNet.

Attributes

An attribute is simply a “numeric tag” attached to either an entity (entity attribute) or a location (location attribute). Since each Cog is inspected for proper spline depth, we attach an attribute called Test to each Cog, specifying the Pass/Fail status of the Cog.

User Distributions

We will sample from a user defined distribution and set the Test attribute to either 1 (for pass) or 0 (for fail). Ninety six percent of the Cogs pass inspection and have their Test attribute set to one. Four percent fail the inspection and have their Test attribute set to zero.

Phased Modeling Approach

Instead of trying to build a model all at once, you may want to implement a phased modeling approach where you build the model in stages. This will help you understand the basic modeling elements before moving on to more complex ones like attributes and IF-THEN logic. The following elements are those components normally entered during each phase of the model building process.

Phase 1: Basic Model Elements

In the first phase you input all of the basic model elements: General Information, Locations, Entities, Arrivals, and Processing. You also import a background graphic to help in placing the locations in the layout window. Upon completion of this phase you have a fully working model, ready to animate and collect output.

Phase 2: Adding Resources & Variability

The second phase normally consists of expanding the model by adding resources and the corresponding path network needed to move entities, using resources, from location to location.

Phase 3: Additional Operations

The final phase usually consists of adding additional reality to the model—that is, adding those nuances that make the model an accurate representation of a real system. These include:

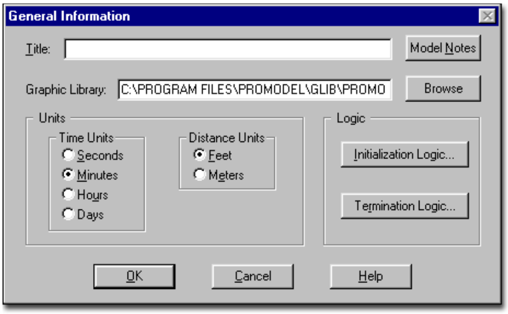
- Adding Quantity and Time_in_system attributes.
- Changing operational times from simple time constants to distributions.
- Adding features that ensure accurate entity processing.
- Adding variables for on-screen display.
- Defining appropriate downtimes.

Phase 1: Basic Model Elements

The first step in building the model is to define the model's basic elements.

General Information

The General Information dialog box, accessed through the Build menu, allows you to name your model and specify default information such as time and distance units. You also specify the name of the graphics library to use. By default it will be PROMOD5.GLB.



General Information

Title:

Graphic Library: C:\PROGRAM FILES\PROMODEL\GLIB\PROMO

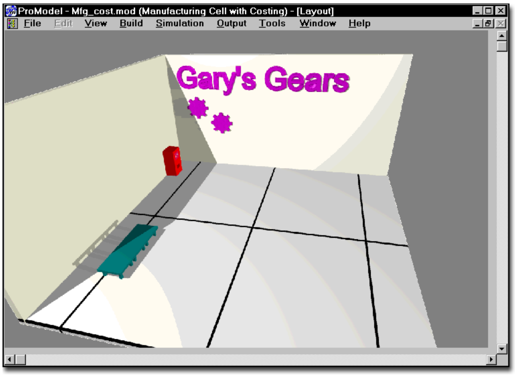
Units

Time Units: ○ Seconds ● Minutes ○ Hours ○ Days

Distance Units: ● Feet ○ Meters

Importing a Background Graphic

Simple background graphics are imported easily through the Background Graphics Editor. ProModel also allows you to import complex graphics files such as AutoCAD drawings to use as the background for your simulations. For more information on how to import a background graphic, see “Graphic Editor” on page 312.



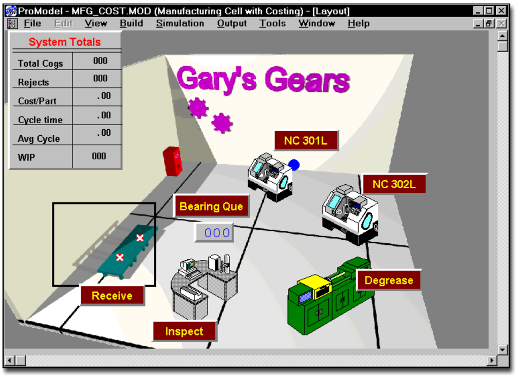
ProModel - Mfg_cost.mod (Manufacturing Cell with Costing) - [Layout]
File
Edit
View
Build
Simulation
Output
Tools
Window
Help
Gary's Gears

Often, importing a background graphic makes the process of placing locations easier, or altogether eliminates the need to create graphic icons for locations.

Defining Locations

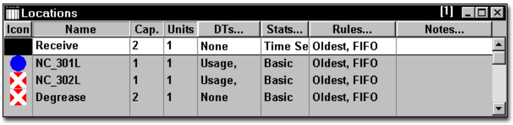
Locations are defined easily by selecting the desired icon and placing it in the layout window. Each time a location is placed in the layout window, a corresponding record is entered in the Location edit table. This table lists each location along with location parameters such as the capacity, number of units, and downtime information. For more information on how to define a location, see "Locations" on page 96.

Layout Window (*maximized*)


ProModel - MFG_COST.MOD (Manufacturing Cell with Costing) - [Layout]
File Edit View Build Simulation Output Tools Window Help
Gary's Gears
NC 301L
NC 302L
Bearing Que
000
Receive
Inspect
Degrease

System Totals	
Total Cogs	000
Rejects	000
Cost/Part	.00
Cycle time	.00
Avg Cycle	.00
WIP	000

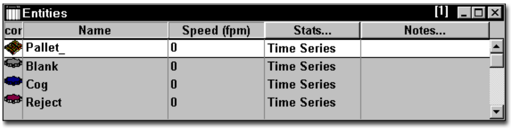
Location Edit Table



Icon	Name	Cap.	Units	DTs...	Stats...	Rules...	Notes...
■	Receive	2	1	None	Time Se	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	

Defining Entities

Once all locations have been defined, we define entities in a similar way by selecting an icon for each entity type. As we do this, a record is created in the Entity edit table for each entity type. For more information on how to define entities, see "Entities" on page 118.

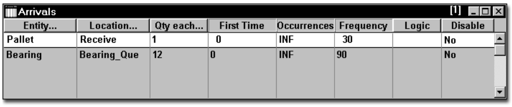


cor	Name	Speed (fpm)	Stats...	Notes...
	Pallet_	0	Time Series	
	Blank	0	Time Series	
	Cog	0	Time Series	
	Reject	0	Time Series	

In this model the Speed (fpm) column is irrelevant since all entities move according to the definition of the mobile resource CellOp. Also, the Stats column shows that we desire detailed statistics for all entity types. "Time series" statistics include throughput history of the entity. "Basic" statistics include only the total exits of each entity type from the system and the final quantity of each entity type in the system.

Defining Arrivals

Of the four entity types, only one needs to be scheduled to arrive in the system. Every 45 minutes one Pallet arrives at the Receiving location. The word "INFINITE" in the Occurrences column means that one pallet continues to arrive every 30 minutes as long as the simulation runs. For more information on defining arrivals, see "Arrivals" on page 163.



Entity...	Location...	Qty each...	First Time	Occurrences	Frequency	Logic	Disable
Pallet	Receive	1	0	INF	30		No
Bearing	Bearing_Que	12	0	INF	90		No

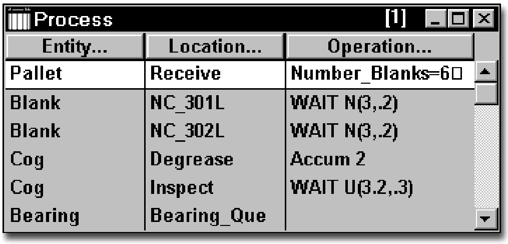
Defining Process Logic

The last step in defining Phase 1 of our model is to define the processing of entities at each location. ProModel simplifies this task by allowing you to select an entity type and then use the mouse to click on the locations in the order in which they will process the entity. Each time you click on a location, a new processing record is added to the Process edit table, defining the process for that entity type at that location. For more information on defining process logic, see “Processing” on page 149, and “Operation Logic” on page 299.

Once the basic entity flow has been defined using the point and click method, operation statements are added to the processing logic. The processing logic can be as simple as a constant operation time or as complex as a nested IF...THEN...ELSE statement.

Process editing actually involves two edit tables that normally appear side by side. The Process edit table specifies what happens to an entity when it arrives at a location, and the Routing edit table specifies where an entity is to be sent once processing is complete.

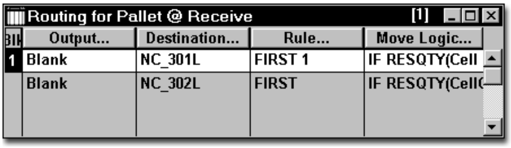
Process Edit Table



Process [1]

Entity...	Location...	Operation...
Pallet	Receive	Number_Blanks=6□
Blank	NC_301L	WAIT N(3,.2)
Blank	NC_302L	WAIT N(3,.2)
Cog	Degrease	Accum 2
Cog	Inspect	WAIT U(3.2,.3)
Bearing	Bearing_Que	

Routing Edit Table



Routing for Pallet @ Receive [1]

Blk	Output...	Destination...	Rule...	Move Logic...
1	Blank	NC_301L	FIRST 1	IF RESQTY(Cell
	Blank	NC_302L	FIRST	IF RESQTY(CellC

Process and Routing Logic

The entire process and routing tables for the Phase 1 model are shown next. The table reads as follows:

1.When an entity called Pallet arrives at location Receive there is no operation time or processing logic (it's just a storage location). The resulting output is six entities called Blank that are routed to the FIRST available destination of either NC_301L or NC_302L.

2.When Blanks arrive at NC_301L or NC_302L, the processing time is a normal distribution with a mean of 3 and a standard deviation of .2 minutes. The name of the entity is now changed to Cog, and the Cog is sent to the Degrease location (FIRST is the default routing rule).

3.Two Cogs are accumulated at Degrease and processed for 5 minutes. When the degrease cycle is complete, Cogs are routed to location Inspect.

4.The inspection time is a uniform distribution with a mean of 3.2 and a half range of .3 minutes. Ninety six percent of the Cogs pass inspection and exit the system, while four percent of the Cogs fail inspection and become Rejects.

➲ Process Table Routing Table

	Location	Operation (min)	Blk	Output	Destination	Rule	Move Logic
Pallet	Receive		1	Blank	NC_301L	FIRST 6	MOVE FOR .5
				Blank	NC_302L	FIRST	MOVE FOR .5
Blank	NC_301L	WAIT N(3,.2)	1	Cog	Degrease	FIRST 1	MOVE FOR .5
Blank	NC_302L	WAIT N(3,.2)	1	Cog	Degrease	FIRST 1	MOVE FOR .5
Cog	Degrease	ACCUM 2 WAIT 5	1	Cog	Inspect	FIRST 1	MOVE FOR .5
Cog	Inspect	WAIT U(3.2,.3)	1	Cog	EXIT	0.960 1	
				Reject	EXIT	0.040	

Phase 2: Adding Resources & Variability

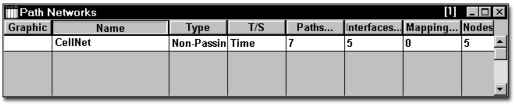
In this phase we wish to add the operator, CellOp, to move entities from location to location, and to perform the loading, unloading, and inspection operations. ProModel requires that all dynamic resources travel on a path network. Therefore, for Phase 2, we need to define resources and path networks.

Defining Path Networks

Path Networks consist of nodes and path segments which connect nodes to other nodes. Each location where a resource may stop to pick up, drop off, or process entities must interface with a path node.

We define path networks through the Path Network edit table similar to the previous edit tables. For each network we specify the nodes and path segments connecting the nodes. Some of the heading buttons, such as Paths and Interfaces, bring up other edit tables such as the Path Segment Edit table shown below. For more information on defining path networks, see “Path Networks” on page 123.

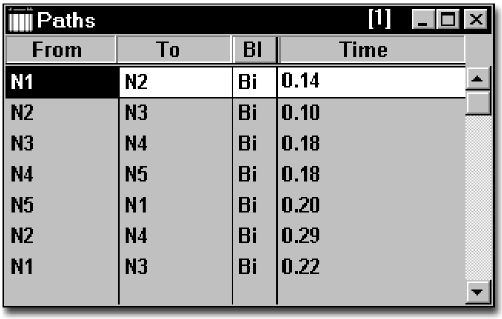
Path Network Edit Table



Path Networks [1]

Graphic	Name	Type	T/S	Paths...	Interfaces...	Mapping...	Nodes
	CellNet	Non-Passin	Time	7	5	0	5

Path Segment Edit Table

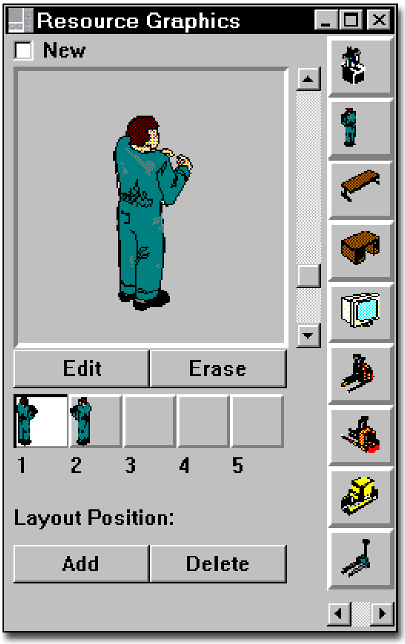


Paths [1]

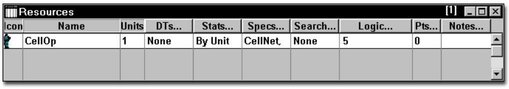
From	To	BI	Time
N1	N2	Bi	0.14
N2	N3	Bi	0.10
N3	N4	Bi	0.18
N4	N5	Bi	0.18
N5	N1	Bi	0.20
N2	N4	Bi	0.29
N1	N3	Bi	0.22

Defining Resources

Resources are defined in much the same way as entities. When in the Resource module we simply select an icon to represent the resource and then specify the characteristics of the resource in the Resource edit table. For more information about defining resources, see “Resources” on page 132.



The Resource edit table shown below contains fields for specifying the name and number of units of a resource. It also has fields for specifying resource downtimes (DTs...), the level of statistics to collect (Stats...), which path network used for travel (Specs...), and any work and park search routines (Search...). Clicking the mouse on any of these buttons brings up separate edit tables for specifying this data.



Resources [1]

Icon	Name	Units	DTs...	Stats...	Specs...	Search...	Logic...	Pts...	Notes...
	CellOp	1	None	By Unit	CellNet,	None	5	0	

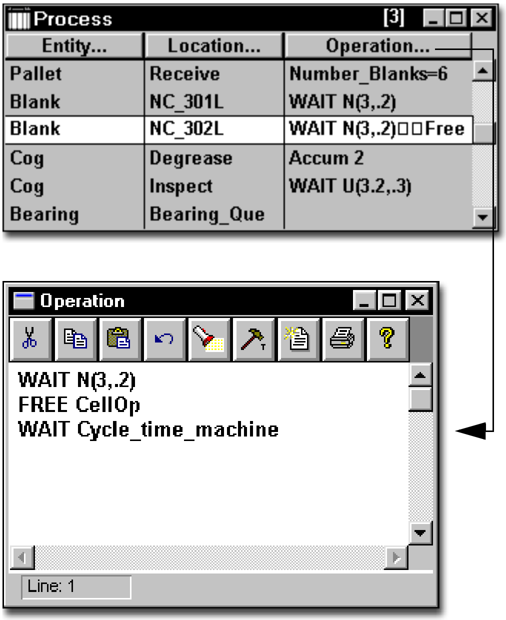
Process Editing

Now that we have defined a resource, we must specify how and when that resource is used in the processing logic.

In the Phase 1 model we used only constant processing times. Now, due to variability associated with the operator, we must represent the loading and inspection times as distributions.

In the example below, CellOp loads the blank at mill NC_302L and is then FREEd to perform other operations. When the Blank has finished processing, the entity is moved with the CellOp to the degreasing machine.

Process Edit Table and corresponding operation logic



Process [3]

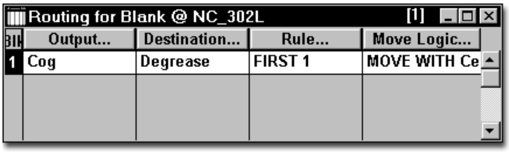
Entity...	Location...	Operation...
Pallet	Receive	Number_Blanks=6
Blank	NC_301L	WAIT N(3,.2)
Blank	NC_302L	WAIT N(3,.2)□□Free
Cog	Degrease	Accum 2
Cog	Inspect	WAIT U(3.2,.3)
Bearing	Bearing_Que	

Operation

```
WAIT N(3,.2)
FREE CellOp
WAIT Cycle_time_machine
```

Line: 1

Routing Edit Table



Routing for Blank @ NC_302L [1]

Blk	Output...	Destination...	Rule...	Move Logic...
1	Cog	Degrease	FIRST 1	MOVE WITH Ce

Process and Routing Logic

The complete process and routing logic is shown below, with CellOp used to perform the loading operations at each mill and inspect the Cogs at the Inspect location. CellOp is also used to transport entities from location to location. All additions or changes to the Phase 1 model are shown in bold type.

Process Table / Routing Table

	Location	Operation (min)	Blk	Output	Destination	Rule	Move Logic
Pallet	Receive		1	Blank	NC_301L	FIRST 6	MOVE **WITH CellOp**
				Blank	NC_302L	FIRST	MOVE **WITH CellOp**
Blank	NC_301L	**WAIT N(3,.2)** **FREE CellOp** WAIT 5.7	1	Cog	Degrease	FIRST 1	MOVE **WITH CellOp**
Blank	NC_302L	**WAIT N(3,.2)** **FREE CellOp** WAIT 5.7	1	Cog	Degrease	FIRST 1	MOVE **WITH CellOp** **THEN FREE**
Cog	Degrease	ACCUM 2 WAIT 5	1	Cog	Inspect	FIRST 1	MOVE **WITH CellOp**
Cog	Inspect	**WAIT U(3.2,.3)** **FREE CellOp**	1	Cog	EXIT	0.960 1	
				Reject	EXIT	0.040	

With the new processing now defined, we have specified all of the necessary modeling elements. We are now ready for the model execution phase. Once again, we shall defer discussion of model execution until we have finished the final phase of the model.

Phase 3: Additional Operations

In the final phase of our modeling session we want to demonstrate an assembly operation by using the operator to install a Bearing into the center hole of the Cog if (and only if) the Cog passes inspection.

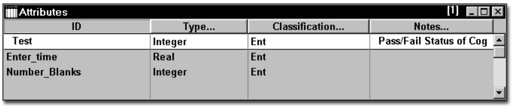
Once we have defined the attribute and distribution table, we must return to the Locations, Entities and Arrivals modules to define a new location called Bearing_Que, a new entity called Bearing, and an arrival schedule for the Bearings.

In addition, we also need to specify a usage based downtime for mills NC_301L and NC_302L from the Location module.

The final step in completing this phase of the model is to edit the processing logic to include the assembly. We will use the built-in JOIN construct to accomplish the assembly.

Defining Attributes

The Attribute module allows you to enter the Attribute edit table to define an Entity Attribute called Test that holds integer values. We set this attribute to one if the Cog passes the inspection, or zero if it fails the inspection. For more information about defining attributes, see “Attributes” on page 225.

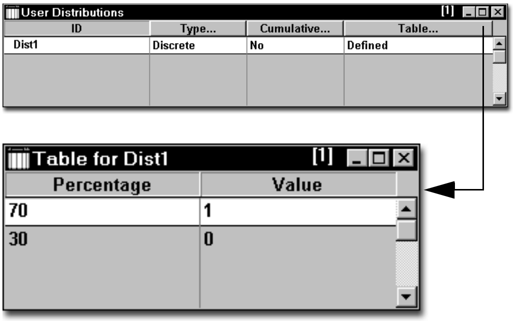


Attributes [1]

ID	Type...	Classification...	Notes...
Test	Integer	Ent	Pass/Fail Status of Cog
Enter_time	Real	Ent	
Number_Blanks	Integer	Ent	

Defining a Distribution

In order to determine if an entity passes or fails the inspection, you sample from a user-defined distribution called Dist1 (alternately, you could use the RAND() function). To define the distribution, simply click the mouse on the Table... button and fill in the distribution parameters. In this case, 96% of the entities pass inspection and 4% fail.



User Distributions [1]

ID	Type...	Cumulative...	Table...
Dist1	Discrete	No	Defined

Table for Dist1 [1]

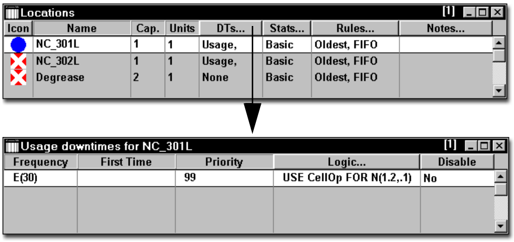
Percentage	Value
70	1
30	0

New Location, Entity, and Arrival

Before we can assemble the Cog at the Inspect location we must first define the new entity type called Bearing in the Entities module. We must also define a new location, Bearing_Que, to hold the Bearings, and an arrival schedule for the Bearings. To do this we simply open the appropriate module as in Phase 1 and supply the information. For simplicity we'll skip the details and move on to downtime specification.

Defining Location Downtimes

In order to represent machine failure times for the two mills, NC_301L and NC_302L, we click on the DT... button in the Location edit table shown below. This brings up another edit table for specifying a downtime based on machine usage. For more information about defining location downtimes, see “Location Downtimes” on page 107.



Locations [1]

Icon	Name	Cap.	Units	DTs...	Stats...	Rules...	Notes...
●	NC_301L	1	1	Usage,	Basic	Oldest, FIFO	
☒	NC_302L	1	1	Usage,	Basic	Oldest, FIFO	
☒	Degrease	2	1	None	Basic	Oldest, FIFO	

Usage downtimes for NC_301L [1]

Frequency	First Time	Priority	Logic...	Disable
E(30)		99	USE CellOp FOR N(1.2,.1)	No

In the example above, we have defined failures to occur according to an exponential distribution with a mean of 30 minutes. When a machine fails, resource CellOp is required to service the machine.

Process and Routing Logic

The process and routing table below shows all of the changes and additions to the Phase 2 model in bold text.

Process Table **Routing Table**

	Location	Operation (min)	Blk	Output	Destination	Rule	Move Logic
Pallet	Receive		1	Blank	NC_301L	FIRST 6	MOVE WITH CellOp
				Blank	NC_302L	FIRST	MOVE WITH CellOp
Blank	NC_301L	WAIT N(3,.2) FREE CellOp WAIT 5.7	1	Cog	Degrease	FIRST 1	MOVE WITH CellOp
Blank	NC_302L	WAIT N(3,.2) FREE CellOp WAIT 5.7	1	Cog	Degrease	FIRST 1	MOVE WITH CellOp THEN FREE
Cog	Degrease	ACCUM 2 WAIT 5	1	Cog	Inspect	FIRST 1	MOVE WITH CellOp THEN FREE
Cog	Inspect	WAIT U(3.2,.3) **Test = Dist1()** **IF Test = 1 THEN** **BEGIN** **JOIN 1 Bearing** **WAIT U(1.2,.2)** **FREE CellOp** **ROUTE 1** **END** **ELSE** **BEGIN** FREE CellOp **ROUTE 2** **END**	1	Cog	EXIT	**FIRST 1**	
			2	Reject	EXIT	**FIRST 1**	
Bearing	**Bearing_Que**		**1**	**Bearing**	**Inspect**	**JOIN** 1	**MOVE FOR .05**

This concludes the final phase of our model building session. We now turn our focus to running the model.

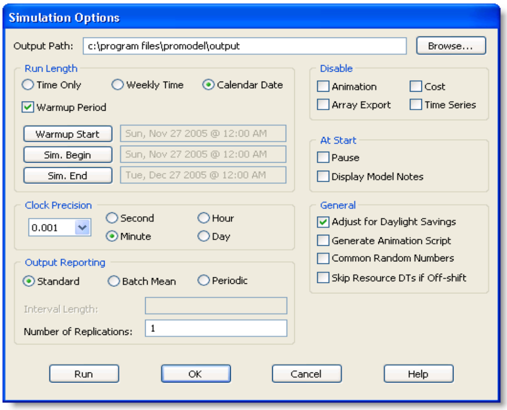
Running a Model

Running a model is a fun and easy process. Models are compiled automatically at runtime, keeping you apart from any complex compilation process. If your model contains any errors, a detailed message explains the nature of the error and points to the module and line number where the error occurred. In most cases you are permitted to make changes on the fly.

ProModel uses concurrent animation, which means that the animation occurs while the simulation is running. Concurrent animation has many advantages over post-simulation animation. By eliminating the two-step process of running the simulation and then animating it, you save valuable time. Concurrent animation immediately allows you to see if a model is working properly.

Simulation Options

When you select Options from the Simulation menu, ProModel displays the Simulation Options dialog. This dialog contains several options for controlling the simulation, such as the run length, warm-up period, clock precision, and the name of the output file. You can also set the number of replications and the level of detail to be collected for the statistics. For more information about Simulation options, see “Simulation Menu” on page 347.



Simulation Options

Output Path: c:\program files\promodel\output [Browse...]

Run Length
○ Time Only ○ Weekly Time ● Calendar Date
☑ Warmup Period
[Warmup Start] Sun, Nov 27 2005 @ 12:00 AM
[Sim. Begin] Sun, Nov 27 2005 @ 12:00 AM
[Sim. End] Tue, Dec 27 2005 @ 12:00 AM

Disable
□ Animation □ Cost
□ Array Export □ Time Series

At Start
□ Pause
□ Display Model Notes

Clock Precision
0.001
○ Second ○ Hour
● Minute ○ Day

General
☑ Adjust for Daylight Savings
□ Generate Animation Script
□ Common Random Numbers
□ Skip Resource DTs if Off-shift

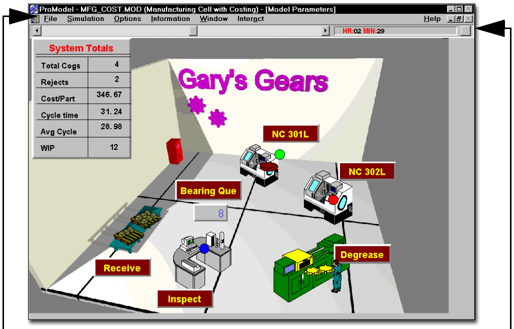
Output Reporting
● Standard ○ Batch Mean ○ Periodic
Interval Length:
Number of Replications: 1

The maximum run length depends on the clock precision and the time unit selected as shown in the following table.

	CLOCK PRECISION			
Time Unit	.01	.001	.0001	.00001
Seconds (sec)	11,930 hrs	1,193 hrs	119 hrs	11 hrs
Minutes (min)	715,827 hrs	71,582 hrs	7,158 hrs	715 hrs
Hours (hr)	42,949,672 hrs	4, 294,967 hrs	429,496 hrs	42,949 hrs
Days (day)	1,030,792,128 hrs	103,079,208 hrs	10,307,904 hrs	1,030,776 hrs

Animation Screen

The ProModel animation screen has a menu of its own, with selections for controlling many simulation parameters (such as run speed). You can also control the animation through panning, zooming, and pausing.



ProModel - MFG_COST.MOD (Manufacturing Cell with Costing) - [Model Parameters]

System Totals	
Total Cogs	4
Rejects	2
Cost/Part	346. 67
Cycle time	31. 24
Avg Cycle	28. 98
WIP	12

Speed Control Bar

Clock Selection Button

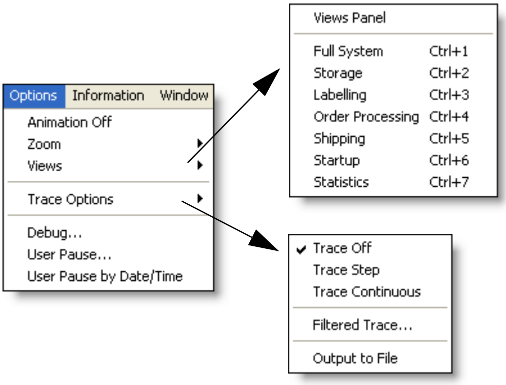
The screen above shows the speed control bar, along with the clock selection button for controlling the format of the clock readout.

The File menu includes an option for viewing a text file of the model as the model is running. This is an excellent way of checking to make sure the model is doing what it is supposed to do!

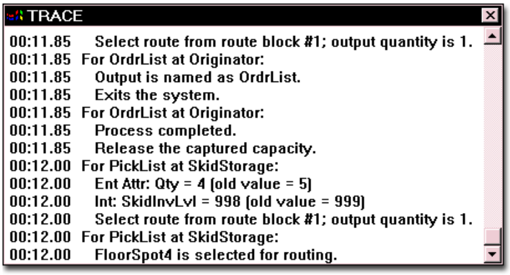
Next we'll take a look at two of the other menu items: Options and Information.

Options Menu

The Options menu contains several options that allow you to track events in the system as they occur. The Debugger is a convenient and efficient way to test or follow the processing of any logic defined in your model. The debugger is used to step through logic one statement at a time and examine variables and attributes while a model is running. A Step Trace allows you to step through the system events one at a time by clicking on the left mouse button. A Continuous Trace allows you to step through system events continuously without clicking the mouse.



The following Trace window shows system events as they occur in the animation. The number in the left hand column represents the simulation time when the event occurred, while the text describes the event. Stepping through the events is an excellent way to verify and debug a model.

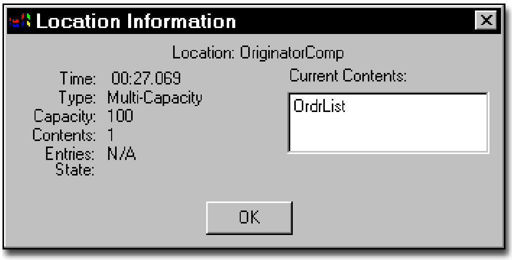


```
00:11.85   Select route from route block #1; output quantity is 1.
00:11.85 For OrdrList at Originator:
00:11.85   Output is named as OrdrList.
00:11.85   Exits the system.
00:11.85 For OrdrList at Originator:
00:11.85   Process completed.
00:11.85   Release the captured capacity.
00:12.00 For PickList at SkidStorage:
00:12.00   Ent Attr: Qty = 4 (old value = 5)
00:12.00   Int: SkidInvLvl = 998 (old value = 999)
00:12.00   Select route from route block #1; output quantity is 1.
00:12.00 For PickList at SkidStorage:
00:12.00   FloorSpot4 is selected for routing.
```

Other options include: Animation Off, which makes the simulation run considerably faster; Zoom, which allows you to zoom in or out to any degree on the animation; Views, which allows you to quickly and easily access specific areas of the model Layout window (see “Layout Settings” on page 86); and User Pause, which allows you to specify the time of the next simulation pause.

Information Menu

The Information menu contains selections for obtaining system information during the run. For up-to-the-minute location information, such as current location contents and total number of entries, select Locations from the Information menu.



Location: OriginatorComp

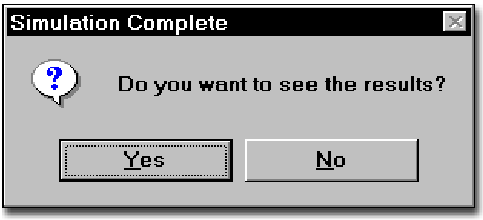
Time: 00:27.069
Type: Multi-Capacity
Capacity: 100
Contents: 1
Entries: N/A
State:

Current Contents:
OrdrList

Viewing Model Statistics & Reports

The purpose of any simulation model is to gain a deeper understanding of the system under study. ProModel's Output Viewer 3DR helps you to see the interactions between various system elements through tabular and graphical representation of system parameters such as resource utilization, throughput history, cycle time, and work-in-process levels.

After each simulation run, you are prompted to view the model output. You can select yes to view the results immediately, or select no to continue with some other task. Selecting yes opens the Output Viewer 3DR and automatically loads the output of the most recent model run.



Simulation Complete

Do you want to see the results?

The Output Viewer 3DR can be run directly from within ProModel, or as an independent application separate from ProModel. You can load a single results file or several results files from different models for comparison of selected statistics.

Model output is written to several output files according to the type of data being collected. The main output file contains information of a summary nature such as overall location utilization and number of entries at each location. Other files keep track of information such as location contents over time and the duration of each entity at each location.

In order to see the power and flexibility of the ProModel output generator, see “Reports and Graphs” on page 373 for detailed examples of the output generated by the Output Viewer 3DR.

Chapter 4: Modeling Environment

The Modeling Environment is everything contained within the ProModel window.

When you open a model or select New from the File menu, your screen appears with a menu bar across the top of the screen and a layout window. You will also be given access to the ProModel Shortcut Panel. For now, let's look briefly at the Shortcut Panel and then the menus accessible from the menu bar.

From the ProModel shortcut panel seen above, you may quickly access some of ProModel’s commonly used features:

- **Open a model** Opens an existing model.
- **Install model package** Loads an existing model package.
- **Run demo model** Allows you to run a demonstration model.
- **www.promodel.com** Immediately connects you with the ProModel support page on the PROMODEL Web site.
- **SimRunner** Launches SimRunner.
- **Stat::Fit** Launches Stat::Fit.

To simply begin working on a new model, close the Shortcut Panel and select New from the File menu. The Shortcut Panel can be opened again from the View menu.

Menu Bar

All of the tools necessary to build and run a model and view the corresponding output are accessed through the menu bar. The menu bar is located just beneath the ProModel caption bar and contains the selections listed on the following page. These selections access other menus with selections related to the menu heading.

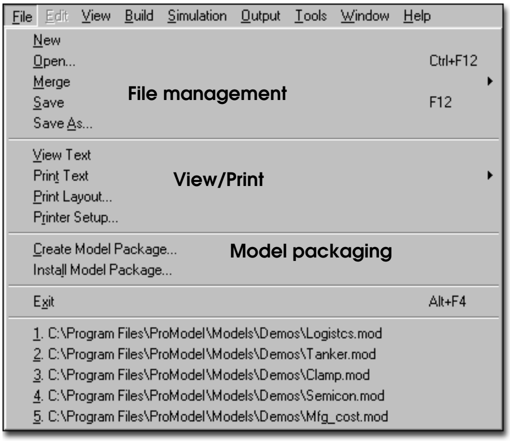
- **File** The File menu allows you to open new models, save current models, and merge two or more models into one. It also allows you to view a text version of the model and print either the model text file or the graphic layout of the model. For more information, see “File Menu” on page 68.
- **Edit** The Edit menu contains selections for editing the contents of edit tables and logic windows. The selections available from this menu will change according to the module from which the Edit menu is selected. They also vary according to the currently selected window. For more information, see “Edit Menu” on page 76.
- **View** The View menu lets you control ProModel’s appearance. From this menu you can control layout settings, hide or view hidden paths, operate the zoom controls, and more. For more information, see “View Menu” on page 83.
- **Build** The Build menu contains all of the modules for creating and editing a model.

This includes the basic modules such as Locations, Entities, Arrivals and Processing, and the optional modeling elements such as Variables, Attributes, Arrays, and Subroutines. For more information, see “Build Menu” on page 95.

- **Simulation** The Simulation menu controls the execution of a simulation and contains options for running a model, defining model parameters, and defining and running scenarios. For more information, see “Simulation Menu” on page 347.
- **Output** The Output selection starts the ProModel Output Viewer 3DR for viewing model output. It also allows you to view the trace generated during run-time. For more information, see “Reports and Graphs” on page 373.
- **Tools** The Tools menu contains various utilities including the Graphics Editor for creating and modifying graphic icons and a search and replace feature for finding or replacing expressions throughout a model. For more information, see “Tools Menu” on page 307.
- **Window** The Window menu allows you to arrange the windows (or iconized windows) that are currently displayed on the screen such that all windows are visible at once. It also allows you to bring any individual window to the forefront of the display. For more information, see “Window Menu” on page 91.
- **Help** The Help menu accesses the ProModel Online Help system. For more information, see “Help Menu” on page 91.

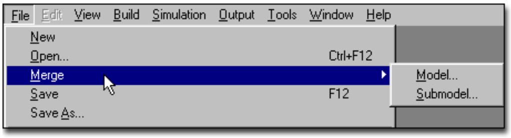
File Menu

The File menu is the first selection on the menu bar and consists of five major sections divided by horizontal lines. The file management section contains functions related to model files such as saving and retrieving. The view/print section allows the user to view a text listing of the current model and print that listing or model layout. The model packaging section allows the user to create and install model packages consisting of models with associated files. Exit quits ProModel, and the model history section lists the five most recently opened models for quick retrieval. Choosing any model in the model history will open and retrieve that model.


File Edit View Build Simulation Output Tools Window Help
New
Open... Ctrl+F12
Merge
Save F12
Save As...
File management
View Text
Print Text
Print Layout...
Printer Setup...
View/Print
Create Model Package...
Install Model Package...
Model packaging
Exit Alt+F4
1. C:\Program Files\ProModel\Models\Demos\Logistcs.mod
2. C:\Program Files\ProModel\Models\Demos\Tanker.mod
3. C:\Program Files\ProModel\Models\Demos\Clamp.mod
4. C:\Program Files\ProModel\Models\Demos\Semicon.mod
5. C:\Program Files\ProModel\Models\Demos\Mfg_cost.mod

File Management

The File menu provides five functions related to model files such as saving and retrieving. Files in the ProModel format use the MOD extension. The following table defines each of the selections available from the file management section shown previously.



New Closes any currently opened model so a new model can be built. This command is unnecessary if no other model is open. If the currently opened model has changed, ProModel will ask if you want to save the model before closing it.

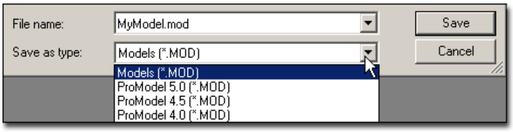
Open Opens a user-specified model and clears previous model data.

Merge Merges a selected ProModel model or submodel into the current model. The same submodel can be merged multiple times into the same model. See “Merge Model” on page 70.

Save Saves an open model under the current file name. If no file name has been given, the user is prompted for a file name.

Save As Saves an open model under a new file name specified by the user. The old file name still exists.

Models saved in current versions of ProModel are not always compatible with previous versions of ProModel.



File name: MyModel.mod

Save as type: Models (*.MOD)

However, models may be saved as previous versions in order to allow those models created in recent versions of ProModel to be shared with others who may be running a previous version of ProModel.

Models created in version 5.4 or higher must be saved as:

- version 5.0 to run with versions 5.0, 5.1, 5.2 or 5.3.
- version 4.5 to run with version 4.5.
- version 4.0 to run with versions 4.0 or 4.2.

Please note

There is also an Autosave feature that saves the model file every n minutes as specified in the **.INI** *file. This feature can be disabled. See the discussion later in this section.*

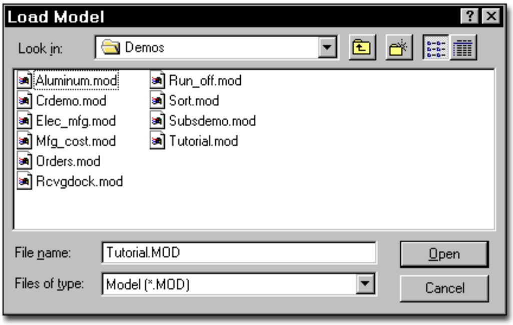
File Management Procedures

How to create a new model:

1. Select **New** from the **File** menu.
2. Define model elements using their corresponding modules.

How to open an existing model:

1. Select **Open** from the **File** menu.
2. Enter the necessary information in the Load Model dialog box.



Load Model

Look in: Demos

File name: Tutorial.MOD

Files of type: Model (*.MOD)

How to save a model:

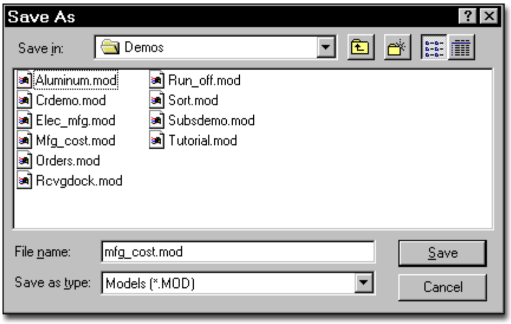
- Select **Save** from the **File** menu. If the model does not already have a name, the Save As dialog box will appear.

How to save a model with a new name:

1. Select **Save As** from the **File** menu.

2. Enter the new file name in the Save As dialog box as shown in the following dialog box.

3. Select **OK**.



Save As

Save in: Demos

File name: mfg_cost.mod

Save as type: Models (*.MOD)

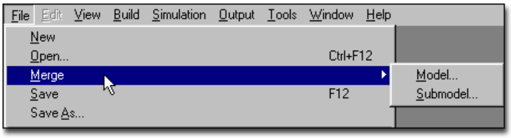
Backup File

ProModel also creates a backup file every time a model is saved. The backup file is named the same as the model file, only with a .BAK extension.

Model Merging

Model merging is a powerful feature that allows large or complex models to be built in smaller segments. A model segment may be as small as a single workstation or as large as an entire department. After all segments are ready, they can be merged together to form a single model.

The Merge feature consists of two options: Merge Model and Merge Submodel.



Merge Model

The Merge Model option allows two or more independent (complete or incomplete) models to be merged into a single model. Duplicate elements found in the base model and the merging model are treated differently according to the element type.

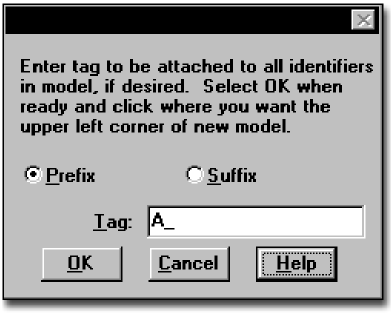
1. Entity and attribute names common to both models are considered common elements in the merged model. For example, if both models contain the entity type In-Box, the merged model will contain only the record from the base model in the Entities table for In-Box.
2. Duplicate locations, resources or path networks *must* first be renamed or deleted from the merging model. Otherwise, an error message occurs and the merge will terminate.
3. If the two models use different graphic libraries, ProModel will give the user the choice to append the merging model's graphic library to the base model's graphic library.
4. All other duplicate model elements cause a prompt to appear with the choice to delete the duplicate element from the merging model or cancel the merge process.

Merge Submodel

The Merge Submodel option allows commonly used submodels to be merged into an existing model in one or more places. Submodels are created just like any other model and may be complete or incomplete models.

When specifying a submodel, you are prompted for a “tag” to be attached to each element of the submodel as either a prefix or suffix. For example, you may be developing a model with four workstations. Instead of creating workstations individually, you could create a submodel with only the common elements (e.g., in-box, out-box, telephone, variables, arrays, etc.) and merge the submodel into the main model four times. In the resulting model, you would then fill in the unique portions of each workstation. Entity and attribute names will not be tagged.

In the following example, the tag “A_” is attached as a prefix to every element of the submodel. A location called Queue1 in the submodel becomes A_Queue1 in the main model and so on. Likewise, a variable called Rejects becomes A_Rejects in the merged model.



Enter tag to be attached to all identifiers in model, if desired. Select OK when ready and click where you want the upper left corner of new model.

● Prefix ○ Suffix

Tag: A_

Please note

Tags used as prefixes must begin with a letter, A through Z, or an underscore. For example, the tag “3C_” is invalid and would produce an error message.

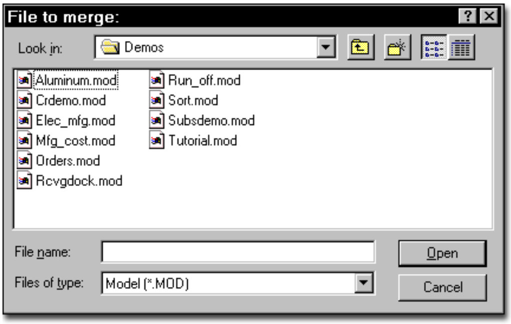
How to merge a model or submodel into an existing model:

1. Open the initial (base) model.

2. From the **File** menu select **Merge**.

3. Select **Model** or **Submodel** from the submenu.

4. Specify the name of the model to be merged in the following dialog box.



File to merge:

Look in: Demos

Aluminum.mod
Crdemo.mod
Elec_mfg.mod
Mfg_cost.mod
Orders.mod
Rcvgdock.mod
Run_off.mod
Sort.mod
Subsdemo.mod
Tutorial.mod

File name:

Files of type: Model (*.MOD)

5. If you select Submodel, specify a prefix or suffix to be attached to each element of the submodel.

6. Click on the layout where you want the model or submodel to appear. A bracket appears on the screen, representing the *upper left corner* of the merging model’s layout. This bracket moves as you move the mouse, allowing you to correctly position the layout to be merged.

7. Next, you will be asked if you would like to append the graphic library file from the model or submodel to the current graphic library file. Select yes or no depending on your preference.

8. When the model is merged in, the graphical elements remain selected so that you can position the merged model exactly where you want it.

Please note

When merging models, if the zoom factors and grid scales are not the same, ProModel will adjust the sizes of graphical elements in the merging model to the scale of the original model.

View/Print Model Text

The modular nature of ProModel makes it easy to focus on the individual elements of a model. However, it can still be useful to see an entire model with all of the model elements in view at one time. ProModel provides two ways to accomplish this. The first is through the View Text option and the second is through the Print Text and Print Layout Options.

The second major division of the File menu contains the following options.

Each menu selection is covered in detail in the following pages.

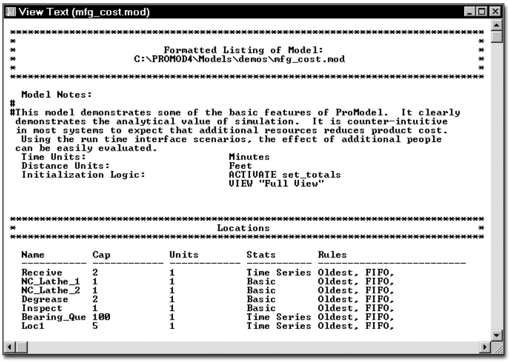
- **View Text** Displays the text of the current model data in a window.
- **Print Text** Prints the text of the current model to either a file or the printer.
- **Print Layout** Prints the model layout to a printer.
- **Printer Setup** Opens a dialog box to allow printers to be selected and controlled.

View Text

The View Text option displays the text of the current model data in a window. This window may be sized or shrunk to an icon for later viewing.

How to view the text of a model:

- Select **View Text** from the **File** menu. The model's text is displayed in a window as shown in the following example.



```
********************************************************************************
*                                                                              *
*                          Formatted Listing of Model:                         *
*                      C:\PROMOD4\Models\demos\mfg_cost.mod                    *
*                                                                              *
********************************************************************************

  Model Notes:
#
#This model demonstrates some of the basic features of ProModel.  It clearly
 demonstrates the analytical value of simulation.  It is counter-intuitive
 in most systems to expect that additional resources reduces product cost.
  Using the run time interface scenarios, the effect of additional people
 can be easily evaluated.
  Time Units:                         Minutes
  Distance Units:                     Feet
  Initialization Logic:               ACTIVATE set_totals
                                      VIEW "Full View"



********************************************************************************
*                                   Locations                                  *
********************************************************************************

  Name        Cap          Units        Stats        Rules
  ----------- ------------ ------------ ------------ -------------------------

  Receive     2            1            Time Series Oldest, FIFO,
  NC_Lathe_1  1            1            Basic        Oldest, FIFO,
  NC_Lathe_2  1            1            Basic        Oldest, FIFO,
  Degrease    2            1            Basic        Oldest, FIFO,
  Inspect     1            1            Basic        Oldest, FIFO,
  Bearing_Que 100          1            Time Series Oldest, FIFO,
  Loc1        5            1            Time Series Oldest, FIFO,
```

Only the first 30 characters of names will appear in the names column.

Please note

You may leave the View Text window open for reference while editing the model. However, any updates will not appear until you close the window and select View Text again.

Print Text

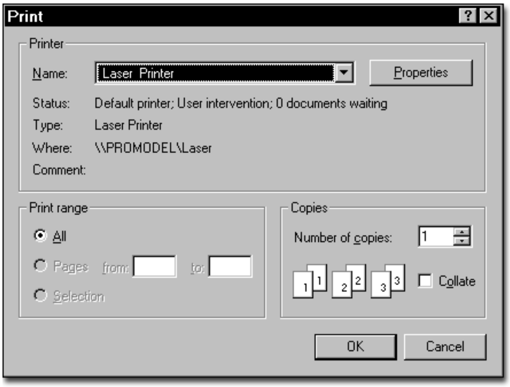
The Print and Print Layout options allow you to print a model to any printer configured for use with Windows. You may also save a text copy of the file to disk.

How to save a text copy of the current model:

1. Select **Print Text...** from the **File** menu.
2. Select **To Text File** from the submenu.
3. Supply a name for the file in the Print to Text File dialog box. The default file extension is TXT.

How to print the current text to a printer:

1. Select **Print Text...** from the **File** menu.
2. Select **To Printer** from the submenu.
3. Select the desired options from the Print dialog box and click **OK**.



Print

Printer

Name: Laser Printer

Status: Default printer; User intervention; 0 documents waiting

Type: Laser Printer

Where: \\PROMODEL\Laser

Comment:

Print range

● All

○ Pages from: to:

○ Selection

Copies

Number of copies: 1

□ Collate

Please note

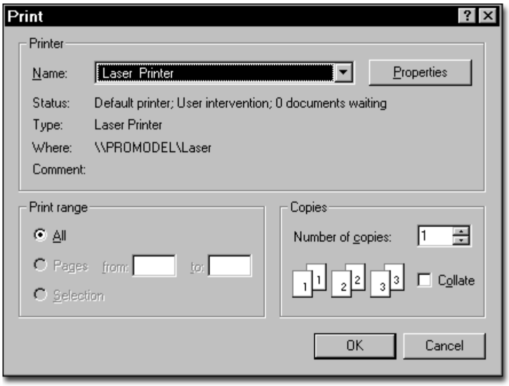
The entire layout may also be copied to the clipboard for editing and printing in another application.

Print Layout

You may print the layout of any model including all locations, path networks, resources, variables and background graphics to any printer configured for use with Windows. Regardless of the size of the model layout, the layout will be proportioned automatically to print on one standard size sheet of paper.

How to print a model layout:

1. Select **Print Layout** from the **File** menu.
2. Select the desired options from the Print dialog box and click OK.



Print

Printer

Name: Laser Printer

Status: Default printer; User intervention; 0 documents waiting

Type: Laser Printer

Where: \\PROMODEL\Laser

Comment:

Print range

● All

○ Pages from: to:

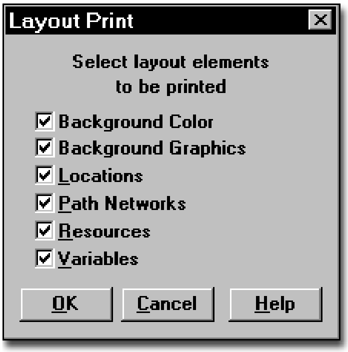
○ Selection

Copies

Number of copies: 1

□ Collate

3. Choose the elements to be included in the layout.



Select layout elements to be printed

☑ Background Color
☑ Background Graphics
☑ Locations
☑ Path Networks
☑ Resources
☑ Variables

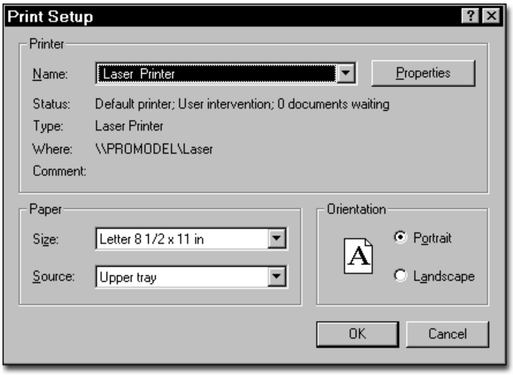
Printer Setup

ProModel allows you to print to any printer configured for use with Windows. At times you may need to switch from the default printer to another printer or plotter. This can be done easily through the Setup option on the print dialog box.

How to change the printer settings:

1. Select **Printer Setup...** from the **File** menu to access the Print Setup dialog box.

2. Select the desired options and click **OK**.



Name: Laser Printer
Status: Default printer; User intervention; 0 documents waiting
Type: Laser Printer
Where: \\PROMODEL\Laser
Comment:

Size: Letter 8 1/2 x 11 in
Source: Upper tray

● Portrait
○ Landscape

Model Packaging/Data Protection

Model packaging and Data Protection are powerful tools that allow you to distribute copies of your model for others to examine and review, yet maintain the integrity of the model. When you create a model package, ProModel builds an archive of files necessary to run the model and allows you to distribute a copy of the model's graphics library. When you apply Data Protection, you can prevent others from viewing or altering logic contained in your model.

Create Model Package Copies the current model and its associated files to a specific directory or disk as <model name>.pkg.

Install Model Package Copies the files in a *.pkg file to the destination directory you wish to use.

Creating a Model Package

The Create Model Package option allows you to copy the current model and its associated files to a specific directory as a single file entitled <model name>.PKG. This file includes the model file (*.MOD), the graphic library (unless you check the Exclude Graphic Library option), and any external files you defined (e.g., read files, arrivals files, and shift files)—the model package automatically includes bitmaps imported into the background graphics.

When you create a model package, two options are available:

- **Exclude Graphic Library** Excludes the graphics library file from the model package—if not required—and creates a smaller package file.

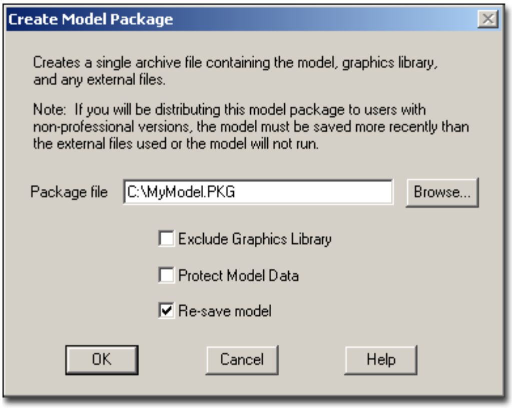
- **Protect Model Data** Prevents those who install the model package from viewing or editing the model data. When you load a model package, ProModel disables the View Text, Print Text, and Expression Search features, plus the Build menu and portions of the run-time Options and Information menus.

Please note

You may NOT use dynamic plots with protected models.

How to create a model package:

1. Select **Create Model Package** from the **File** menu.



Create Model Package

Creates a single archive file containing the model, graphics library, and any external files.

Note: If you will be distributing this model package to users with non-professional versions, the model must be saved more recently than the external files used or the model will not run.

Package file: C:\MyModel.PKG

□ Exclude Graphics Library

□ Protect Model Data

☑ Re-save model

2. Enter the name you wish to use for the model package (by default, ProModel uses the name of the current model with a ***.pkg** extension). You may also use the **Browse...** button to select the model name and directory.

3. Check the **Exclude Graphics Library** box if you want to package the model *without* the graphics library.

4. Check the **Protect Model Data** box if you want to protect your model data and prevent other users from changing or viewing the model data.

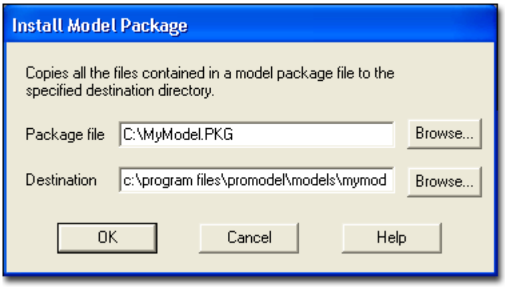
5. Click **OK**.

Installing a Model Package

Install Model Package copies all files in a model package to a specified destination directory and gives you the option to load the model.

How to install a model package:

1. Select **Install Model Package...** from the **File** Menu.



Install Model Package

Copies all the files contained in a model package file to the specified destination directory.

Package file: C:\MyModel.PKG

Destination: c:\program files\promodel\models\mymod

2. Select the model package (*.pkg) from the Install Model Package dialog.

3. In the Destination field, type the name of the directory to which you want to copy the model package.

4. Select **OK**. After you install the model package, a dialog will appear and allow you to load the model.

Edit Menu

The Edit menu is the second selection on the menu bar. It changes forms depending on which window is currently active when choosing Edit.

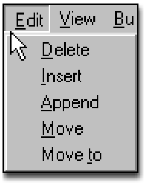
- If the active window is an edit table such as the Locations edit table, the Edit menu contains options for inserting, deleting and moving records.
- If the active window is the Processing edit table, the Edit menu contains two additional options for copying and pasting completed process and routing records.
- If the active window is a Notes window, the Edit menu contains selections for cutting, copying, and pasting text within the current window or transferring it to another window.
- If the active window is a logic window, such as the one for the operation column of the processing module, the Edit menu contains an additional selection, Compile, which checks the syntax of the logic in the logic window. This option is additional to the cut and paste functions normally available in a notes window.
- Finally, if the active window is either the Graphic or Background Editor, the Edit menu contains three sections. The first section has choices for cutting, copying, and pasting graphic objects to and from Pro-Model's internal clipboard. The second has options to cut and paste items to and from the Windows Clipboard. The third has options to import and export graphics from other applications.

Please note

The Edit menu is accessible only while working inside a module such as a processing edit table or a Notes edit window. If no module is currently open, the Edit selection is not available. Because multiple edit tables and windows may appear on the screen at the same time, the Edit menu commands pertain only to the currently active window. (To activate a window, click anywhere inside it.)

Editing Tables

When creating or modifying records in an edit table such as a Locations or Entities table, the Edit menu appears as follows.



The following table briefly lists the function of each selection of the Table Edit menu.

Delete Deletes a record from the table.

Insert Inserts a record in the table above the current record.

Append Appends a record to the end of the table.

Move Marks a record for moving to a new position in the table. Only one record may be marked at a time.

Move to Moves the previously marked record in the table above the current record.

How to delete a record from a table:

1. Select the desired record by clicking in any field of the record.
2. Select **Delete** from the Table **Edit** menu.

How to insert a record in a table:

1. Position the cursor in the record *below* where you wish the new record to be inserted.

2. Select **Insert** from the Table **Edit** menu.

How to append a record to the end of a table:

1. Position the cursor in any record of the table.

2. Select **Append** from the Table **Edit** menu. A new record will appear below the last record of the table.

How to move a record to a new position in a table:

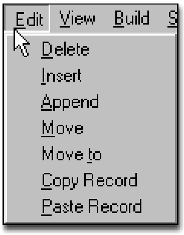
1. Select the record to be moved by placing the cursor in any field of the desired record.

2. Select **Move** from the Table **Edit** menu.

3. Position the cursor in the record that is *below* the final destination of the selected record and select **Move to** from the Table **Edit** menu.

Editing Process Records

When editing the records in the Processing edit table, two additional options for copying and pasting entire process and routing records appear in the Edit menu.



The Processing Edit menu's first five selections are identical to the Table Edit menu. The two additional menu items are as follows:

Copy Record   Copies all fields of the current record for subsequent pasting.

Paste Record   Places a copy of the most recently copied record above the current record.

Please note

Notes on editing process records:

1. A process record consists of all fields in the Process edit table, as well as all corresponding routing records defined in the Routing edit table for the given process.

2. From a Routing edit table, you may only use the Copy Record and Paste Record options to copy routing information to another routing record.

3. These options cannot be used to copy processing or routing records from one model to another.

How to copy information from another record:

1. Select the information to copy from field of the desired record.

2. From the **Edit** menu, select **Copy Record** (or press CTRL + C) to copy the information.

3. In the field you want to place the information, select **Paste Record** from the **Edit** menu (or press CTRL + V).

Please note

Alternately, you may copy information between records using the right-click menu.

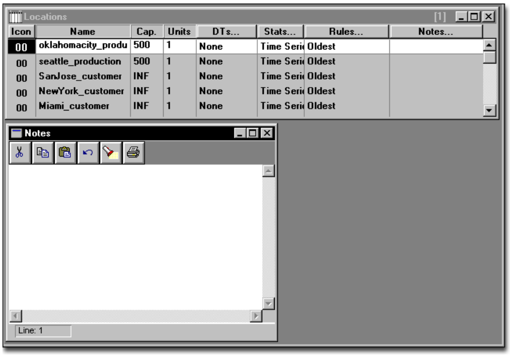
Editing Notes Windows

You may annotate individual records of the locations, resources and entities tables through Notes windows. When editing the text of a notes window, the Edit menu changes to the following form.

How to annotate a record:

1. Click inside the record to be annotated.

2. Click on the Notes button at the right of the table.



Icon	Name	Cap.	Units	DTs...	Stats...	Rules...	Notes...
00	oklahomacity_produ	500	1	None	Time Seri	Oldest	
00	seattle_production	500	1	None	Time Seri	Oldest	
00	SanJose_customer	INF	1	None	Time Seri	Oldest	
00	NewYork_customer	INF	1	None	Time Seri	Oldest	
00	Miami_customer	INF	1	None	Time Seri	Oldest	

The selections available from the Notes Edit menu are defined as follows.

Cut Removes the selected text and places it in the clipboard.

Copy Copies the current text and places it in the clipboard.

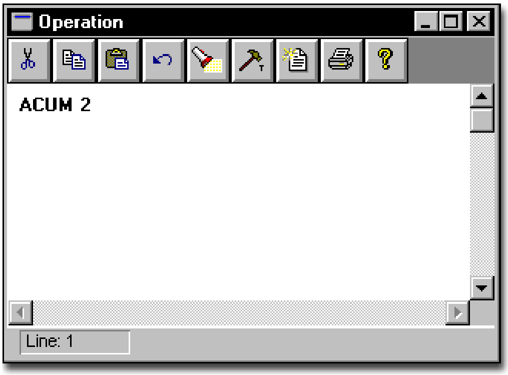
Paste Inserts the contents of the clipboard at the cursor.

Clear Deletes the selected text *without* placing a copy in the clipboard.

In addition to the options in the edit menu, the notes window itself contains four buttons. There are three edit buttons, Cut, Copy, and Paste which work exactly the same as the corresponding options in the Edit menu, and a Print button. The print button prints the text in the notes window. A status bar appearing at the bottom of the Notes edit window shows the current line position of the cursor (e.g., Line: 2).

Editing Logic Windows

All multi-line logic windows, such as the operation logic window of the Process edit table, include the editing function buttons shown in the following example. These buttons include cut, copy, paste, undo, local find and replace, build, compile, print, and help. Using the Cut, Copy, or Paste button works exactly the same as using the corresponding option from the Edit menu.

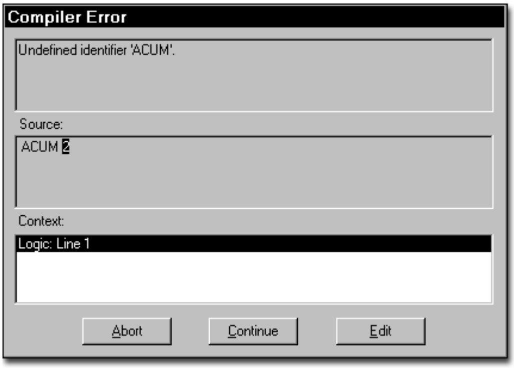


```
ACUM 2
```

Line: 1

Build The Logic Builder, which may also be accessed by clicking the right mouse in the logic window button, is a tool that allows you to build logic without typing a single keystroke. Primarily for building logic in the logic windows, it may also be used for building expressions in expression fields.

Compile The Compile menu item checks the logic in the edit window to see if it is complete and syntactically correct. If an error is found, an information box with details regarding the error(s) appears. For example, the following error is a result of the incorrectly spelled statement "Acum 2."



Undefined identifier 'ACUM'.

Source:

```
ACUM 2
```

Context:

Logic: Line 1

Print Prints text, such as the operation logic for a single location, in a logic window and is helpful when you are trying to debug a model and need to work with particularly complex logic at certain locations.

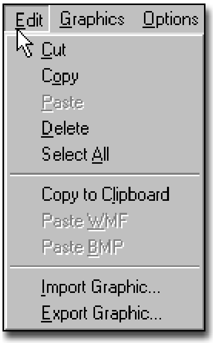
Help Provides context-sensitive help. ProModel will provide help with select words or any word with the cursor next to it.

Please note

A status bar is displayed at the bottom of a Logic edit window, and shows the current line position of the cursor (e.g., Line: 2).

Editing Background Graphics

When working with graphic objects in the Background Graphics module, the Edit menu appears as follows.



Cut Removes the selected object(s) and makes a temporary copy that may be pasted back into the layout window.

Copy Makes a temporary copy of the selected object(s) for pasting later.

Paste Adds the most recently cut or copied object(s) to the layout window.

Please note

While in the Background Graphics module, ***CUT****,* ***COPY****, and* ***PASTE*** *all use* ProModel*'s internal clipboard. Any objects copied to this clipboard cannot be pasted into other Windows applications. To copy objects to other applications, use* ***COPY TO CLIPBOARD****.*

Delete Deletes the selected objects from the layout window.

Select All Selects all of the objects in the layout window.

Copy to Clipboard Copies the *entire* contents of the workspace to the Windows clipboard. Objects cannot be copied individually to the clipboard. ProModel copies the workspace as a bitmap or windows metafile for easy transfer to other graphics packages that use the Windows clipboard.

Paste WMF Pastes a Windows metafile (WMF) from the Windows clipboard into the Edit 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 Edit window. You must have previously copied a bitmap to the Windows clipboard.

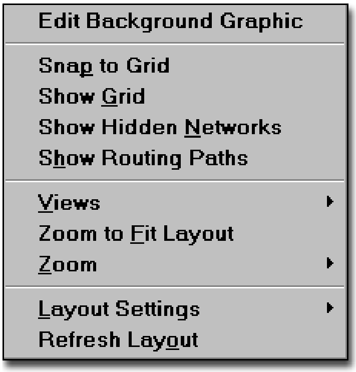
Import Graphic Imports a WMF, BMP, PCX or GIF file into the layout window.

Export Graphic Exports the graphic in the layout window to a WMF or BMP file.

How to edit background graphics:

1. Right click on the background graphic you wish to edit.

2. Select **Edit Background Graphic** from the right-click menu.



3. Edit the graphic as desired.

Editing & Moving Graphics

ProModel allows easy adjustment and fine tuning of the graphics in one window rather than opening individual modules to move each object. This makes it possible to move or rearrange a whole submodel or model after it has been merged. To move any or all graphics in a model, ProModel must be in ***common mode*** (all modules must be closed, leaving only the layout window open)**.**

While in common mode, graphics that can be moved and arranged in the layout window include locations, path networks, static and dynamic resources, variables, and background graphics. Multiple graphics can be selected and moved simultaneously. When graphics are selected, the layout can be scrolled and the graphics will remain selected.

How to move graphics:

1. Close all build modules leaving only the layout window open.

2. Left click and drag the selected graphic to the desired position.

3. To move multiple graphics, hold down the Shift key while left clicking the desired graphics to select them and then drag them to the desired position. Using the shift key with a left click deselects a selected graphic.

4. Fine tune the position of the selected graphic(s) by using the arrow keys on the keyboard to move one pixel at a time.

When a submodel or model is merged into another model, all graphics associated with it are selected. This allows for immediate movement by left clicking on the graphics and dragging them to the desired position in the Layout window. You may also scroll the layout first and then move the selected graphics.

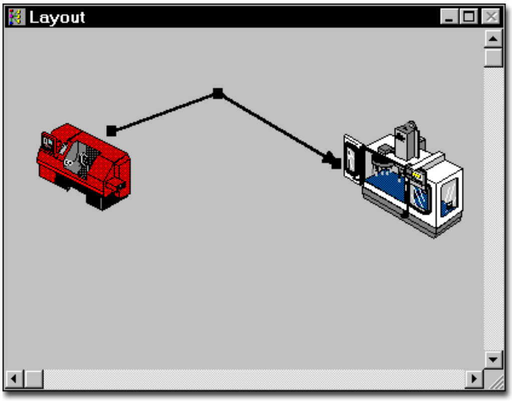
Please note

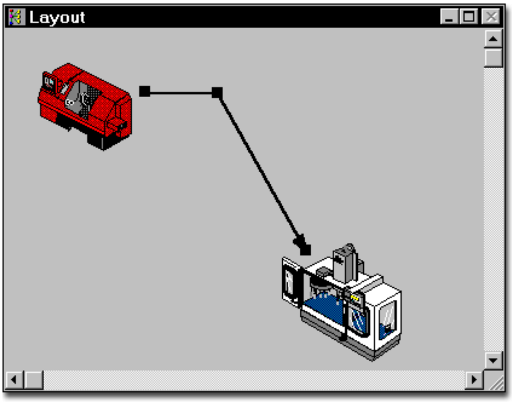
Multiple graphics can also be selected by dragging a bounding box around the graphics. Holding the left mouse button down, drag the mouse from one corner of the graphics to the opposite corner. The bounding box must completely enclose the graphics you wish to select.

Special Considerations for Moving Graphics

Locations   When you click on a location with multiple graphics defined for it (i.e., counter, status light, label) in common mode, it is selected as one graphic. Multiple units of a location can be moved individually. Adjusting the position of an individual graphic in a multiple graphic location must be done in the locations module.

Routing Paths   Routing paths defined in the Processing module cannot be selected for movement but will move when a location to which they are connected is moved. If a routing path has multiple segments, only the segment connected to the location being moved gets readjusted. However, when both locations on the routing path are moved simultaneously, the whole routing path is moved.





In the examples above, the machine on the right is moved. The routing paths remain connected to each graphic while the path segment connected to the moved graphic is adjusted accordingly.

Path Networks Path networks are not resized when moved. Any resource points defined for a dynamic resource are moved when the path network is moved. The resource points remain relative to the node for which they were defined.

Resources When a dynamic resource is moved, the resource point associated with it moves but not the entire path network.

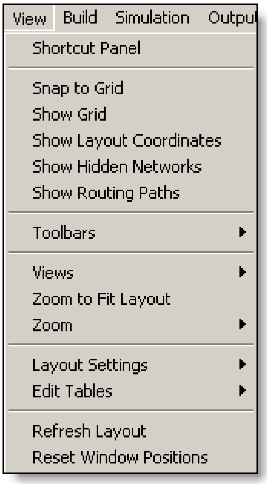
Sizing Graphics cannot be sized in common mode.

Snap-to-Grid Graphics will not snap to the grid in common mode, but their position can be fine-tuned using the arrow keys on the keyboard to move the selected graphic(s) one pixel at a time.

Labels To view the label or edit a location, path network, resource, or variable graphic, right click on the item. If you double left click on an item, ProModel opens the build module and highlights the record where you defined the element.

View Menu

The View menu provides options for modifying the model editing environment. These options are defaults, used each time the program is started and are not specific to any particular model. The View menu consists of three categories: Switches, Settings, and Commands. Each of these categories is explained in the following sections.

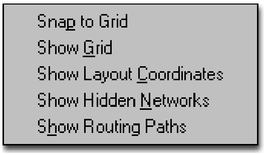


How to access the View menu:

- Select **View** from the menu bar.

Switches

The following selections are available from the switches section of the View menu. Switches are options you can check or uncheck to turn on or off.



Snap to Grid Check this switch to cause any object subsequently drawn or placed on the layout to be positioned on the nearest grid line. Snap to grid snaps the upper left corner of a graphic.

Show Grid Check this switch to show the grid in the layout window.

Show Layout Coordinates Displays cursor's the coordinates in the upper-left-hand side of the layout window.

Show Hidden Networks Check this switch to show the invisible path networks during editing. A path network can be made to be invisible by selecting the "invisible" option for the particular network while in the Path Networks editor.

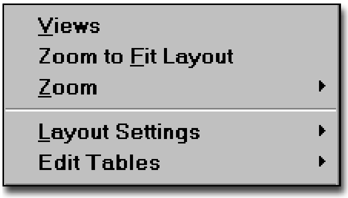
Show Routing Paths Causes routing paths to be visible during run time as well as edit time. If the option is not checked, the routing paths are visible only while in the processing editor.

Toolbars

Allows you to toggle the various ProModel toolbars on and off. See "Toolbars" on page 92 for more infomation on each toolbar.

Settings

The following table defines each of the selections available from the settings section of the View menu. These selections and their submenus are discussed in more detail on the following pages.



Views Allows you to define, then quickly and easily access specific areas of the model layout. Once the view is defined, you can select it while editing or running the simulation.

Zoom to Fit Layout Shrinks or enlarges the layout to include the entire model.

Zoom Allows you to shrink or enlarge the layout by the percentage selected.

Views

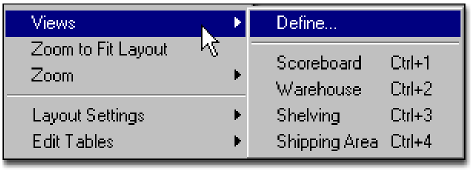
The Views feature allows you to define, then quickly and easily access specific areas of the model layout. Selecting a view scrolls the layout window and adjusts the zoom so you see a specific region of the layout regardless of the layout window's size. Once a view is defined, you can select it while editing or running the simulation by selecting the view from the View menu or by using the keyboard shortcut.

Defining & Selecting Views

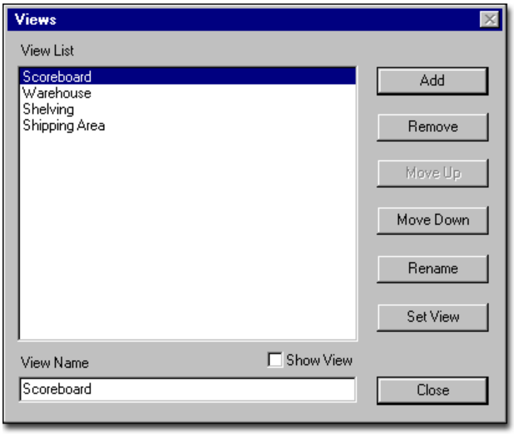
How to define a view:

1. At edit time, select the **View** menu.

2. Select the **Views** item. If no defined views exist, the Views dialog is displayed and not the submenu shown here. If the Views dialog appears, skip step 3.



3. Click **Define** from the extended menu and the Views dialog appears.



View Name: Scoreboard

4. With the Views dialog open, select the area in the layout window you want to define as a view using the scroll bars and zoom feature in the View menu. Or size the layout window to the desired view.

5. With the layout window set, type a name for your view in the View Name field at the bottom of the Views dialog, and then click the **Add** button to define the view you have selected in the current layout window.

6. The name you have entered will appear in your list of views.

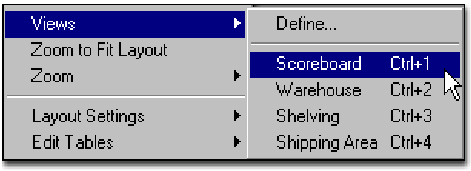
Please note

Views cannot be defined at run time.

How to select a view from the menu:

1. At edit time or run time, select the **View** menu.

2. Select the **Views** menu item to display the submenu list.



3. Click on the desired view from the submenu.

How to select a view with shortcut key CTRL + n:

- Press CTRL + *n* to select the desired view where *n* is the number (1-9) of its position in the view list (e.g., pressing CTRL + 1 would access Single Screen Zoom from the view list above and CTRL + 3 would access the Waiting Time Zoom view). Views beyond nine will not have a CTRL + *n* shortcut key.

Managing Your Views

When you click on Define from the Views menu, the Views dialog is displayed. It lists the defined views in the Views List and provides buttons for adding views, removing views, and managing the list. The function of each button is described next.

Add Adds the view, which you have named in the View Name field, to the list of views.

Remove Deletes the currently highlighted view from the View List.

Move Up Moves the currently highlighted view up one position in the list creating a corresponding change to the menu and CTRL+ *n* order.

Move Down Moves the currently highlighted view down one position in the list creating a corresponding change to the menu and CTRL+ *n* order.

Rename Renames the currently selected view to the name you have typed into the View Name field.

Set View Sets the highlighted name in the View list to the portion of the model currently visible in the Layout window.

View Name This is the field where you type the name of the view you wish to add.

Show View Checking this box will cause your views to be displayed in the Layout window as you select them in the list.

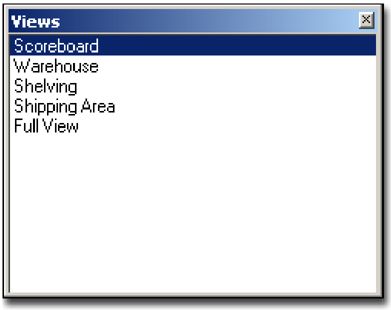
Using Views at Run-time

When the simulation is running, you may choose the active view by picking its name from the Views menu or the Views Panel.

The Views menu is accessible during run-time from either the Menu Bar's Option menu or the Right-Click menu.

The Views menu will display a list of your defined views. There is also an option to open the Views Panel.

The Views Panel is a dialog window with a list of your defined views.


Views

Scoreboard
Warehouse
Shelving
Shipping Area
Full View

The Views Panel may be moved or resized, and will remain on top of the simulation window until the Panel is closed.

Simply click on the name of a view in the Panel to switch to that view.

Referencing a View in Model Logic

Once a view has been defined, it may be referenced in the model using the VIEW statement (e.g., VIEW “Service Office”). This is useful for illustrating certain parts of the model at specific times during run-time. For syntax and examples, see “View” on page 576.

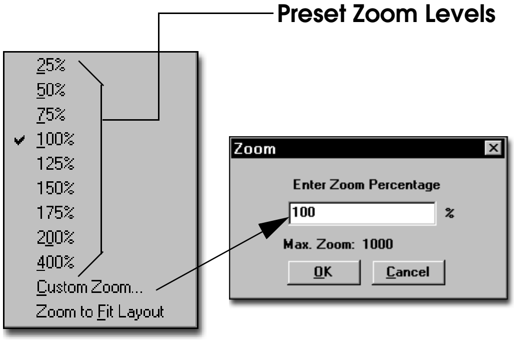
Zoom Feature

The Zoom feature allows you to shrink or enlarge the layout by the percentage selected.

How to zoom in or out on the layout:

1. Select **Zoom** from the **View** menu.

2. Choose any preset zoom level, or the **Custom Zoom** option.



Enter Zoom Percentage: 100 %

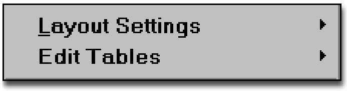
Max. Zoom: 1000

Please note

The minimum and maximum zoom levels are calculated automatically depending on the total size of the layout at 100% zoom.

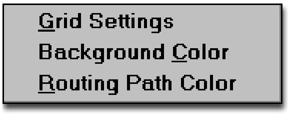
Layout Settings

The following selections are available from the Settings section of the View menu.



The Layout Settings submenu contains selections for changing the grid characteristics, color of the layout window background, and the routing arrow colors. These, as well as default setting changes, apply to any currently loaded model. To change any of these items for the current model only, use the options provided in the General Information dialog. The routing path color can be changed for the current model only by selecting the Path Options button in the Processing module.

The following table defines each of the selections available from the Layout Settings submenu.



Grid Settings Provides options to control the amount of space between grid lines. It also provides the option to define the grid units in terms of distance and time per grid unit.

Background Color Allows the user to change the background color in the layout window.

Routing Path Color Provides the option to change the routing color used in processing logic.

Please note

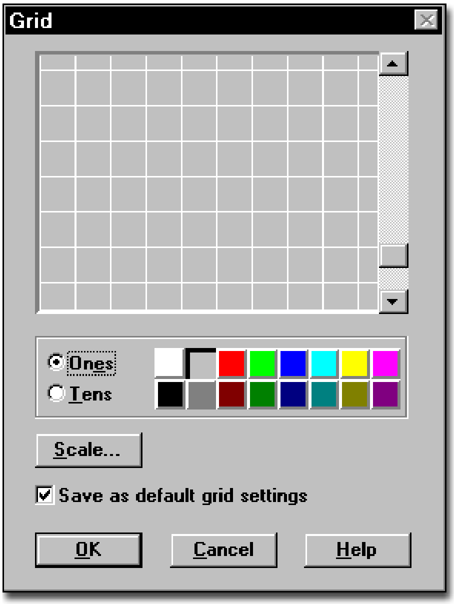
Changes to these settings are saved as the default settings.

Grid Size

By using the grid dialog box, you may set the resolution of the grid lines to your preference.

Please note

You may save the grid settings for the model by checking the Save as default grid settings check box.



● Ones
○ Tens

☑ Save as default grid settings

How to change the colors and resolution of the grid lines:

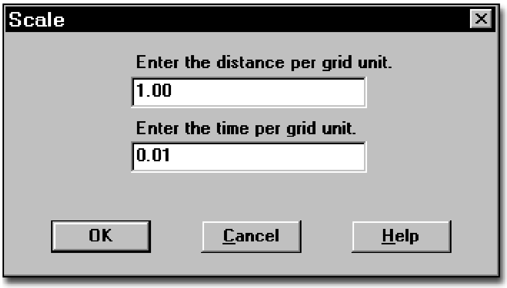
1. Select **Layout Settings** from the **View** menu.
2. Select **Grid Settings** from the **Layout Settings** submenu.
3. Select the **Ones** or **Tens** option button.
4. Select the desired color.
5. Use the scroll bar to adjust the resolution.

Please note

To change the color of the grid lines, select the Ones button and choose a color. To change the color of every tenth grid line, select the Tens button and choose a color.

Grid Scale

In addition to setting the resolution of the grid lines, you may also associate a time and distance value to each grid unit. This is extremely useful when you are creating conveyors, queues, or path networks to scale and you want the time or distance between nodes to be based on the number of grid units between the nodes.



Enter the distance per grid unit.
1.00

Enter the time per grid unit.
0.01

How to set the default time and distance per grid unit:

1. Select **Layout Settings** from the **View** menu.
2. Select **Grid Settings** from the **Layout Settings** submenu.
3. Select the **Scale** button from the Grid Dialog.
4. Enter the desired time and distance per grid unit.

Please note

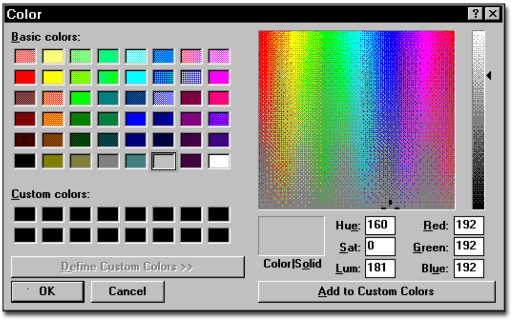
The "Recalculate path lengths when adjusted" option applies to path networks, conveyors, and queues. For details regarding recalculation of times and distances when editing path segments, see "Path Networks" on page 123.

Background Color

The Background Color option allows the user to change the background color in the layout window.

How to set the background color of the layout:

1. Select **Layout Settings** from the **View** menu.
2. Select **Background Color** from the **Layout Settings** submenu.
3. Select the desired color.
4. Click **OK**.



Hue: 160
Sat: 0
Lum: 181
Red: 192
Green: 192
Blue: 192

Custom Colors

ProModel allows you to create up to 16 custom colors for use anywhere a color selection is available. Entity, location, and resource icons, as well as background graphics, can use any custom color defined in the colors menu.

When creating a custom color, the nearest solid color is shown next to the dithered color for reference.

How to create a custom color and add it to the color menu:

1. Select **Layout Settings** from the **View** menu.

2. Select **Background Color...** from the **Layout Settings** submenu.

3. Move the cursor to the area on the multi-color chart closest to the custom color you desire and click the left mouse button.

4. Adjust the color by moving the custom color adjustment slider up or down until the desired shade is obtained. Alternately, you can manually adjust any of the color definition fields (Hue, Sat, Lum, Red, Green, Blue).

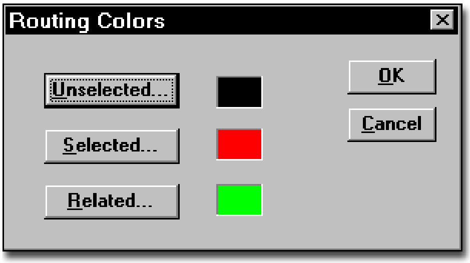
5. Select **Add to Custom Colors**. The color now appears in one of the 16 custom color boxes.

Routing Path Color

ProModel allows you to select which colors to use when showing selected, unselected and related routings in the Processing module. This helps in visually identifying the origin and destination of a process routing.

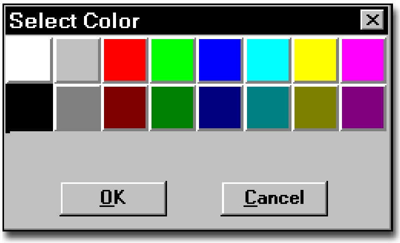
How to specify the routing colors:

1. Select **Routing Path Color** from the **View** menu.



2. Click on the desired routing type, **Unselected**, **Selected**, or **Related** from the menu.

3. Select the desired color.



4. Click **OK**.

Routing Path Types

Unselected All routing lines *not* for currently highlighted process record.

Selected Routing line for currently highlighted routing record.

Related Routing lines for highlighted process record, except highlighted routing record.

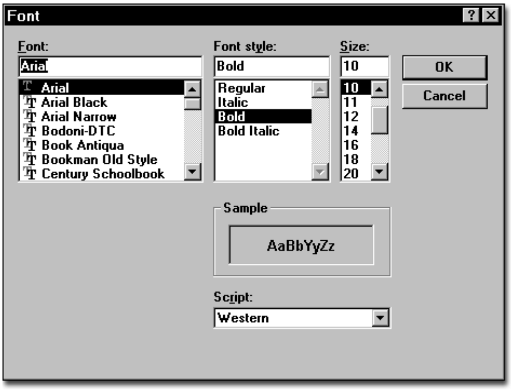
Edit Tables

Edit Table Fonts

Edit tables are used extensively in ProModel for data entry. ProModel allows you to specify the font used in these tables.

How to change the edit table font:

1. Select **Edit Tables** from the **View** menu.
2. Select **Font** from the **Edit Tables** submenu.



Font: Arial
Font style: Bold
Size: 10
Sample: AaBbYyZz
Script: Western

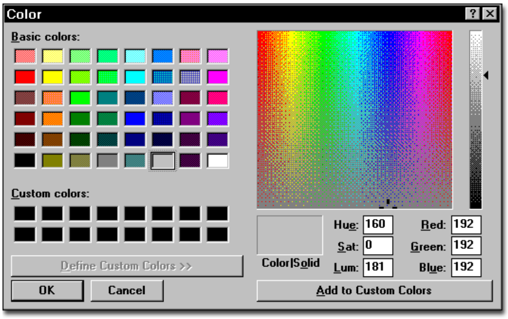
3. Choose the desired font by scrolling through the Font selection list box.
4. Choose the Font Style.
5. Choose the Font Size.
6. Click **OK**.

Edit Table Color

ProModel allows you to specify the table color used in edit tables.

How to change the edit table background color:

1. Select **Edit Tables** from the **View** menu.
2. Select **Color** from the **Edit Tables** submenu.
3. Choose the desired color.
4. Click **OK**.



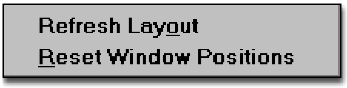
Hue: 160 Sat: 0 Lum: 181
Red: 192 Green: 192 Blue: 192

Please note

For information on creating custom colors, see "Background Color" on page 88.

Commands

The Commands section of the View menu contains various selections for controlling the modeling environment.

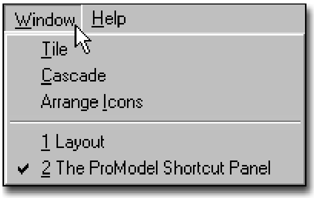


Refresh Layout Clears and redraws the graphics in the layout window.

Reset Window Positions Causes all edit tables to return to their original positions and sizes.

Window Menu

The Window menu allows you to rearrange windows and icons and select the active window. These functions are standard to all Windows™ applications.



Tile Causes all open windows to fit in the available screen space. Windows that may be hidden behind other windows become visible.

Cascade Causes all open windows to overlap such that the title bar of each window is visible.

Arrange Icons Causes all icons representing iconized applications to be arranged neatly along the bottom of the screen.

Please note

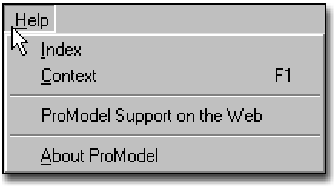
Below **Arrange Icons** *is a list of the open windows. The window with the check next to it is the active window.*

How to reset the windows to their default positions:

- Choose **Reset Window Positions** from the **View** menu.

Help Menu

The ProModel Help menu is a convenient, quick way to look up information about a task you are performing, a feature you would like to know more about, or a command you want to use. ProModel Help is available whenever you see a Help command button, or Help as an item on a menu bar.



The selections from the Help Menu are as follows:

Index Brings up the Main Help Index.

Context Opens the help system to the topic related to the currently active window.

ProModel Support on the Web When you select this option from the help menu, ProModel automatically connects you with the ProModel customer service page on the PROMODEL web site.

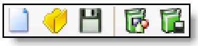
About ProModel Displays a message containing information about the product.

Toolbars

The ProModel toolbars provides quick access to many of the options found in the menu bar.

The toolbar space may contain up to nine toolbars, which can be be toggled on or off in the View menu. These toolbars are described below.

File



New Closes any currently opened model so a new model can be built. This command is unnecessary if no other model is open. If the currently opened model has changed, ProModel will ask if you want to save the model before closing it.

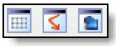
Open Opens a user-specified model and clears previous model data.

Save Saves an open model under the current file name. If no file name has been given, you will be prompted for a file name.

Create Package Bundles all files used for a model into one package. See “Creating a Model Package” on page 74 for more information.

Install Package Copies all files in a model package to a specified destination directory and gives you the option to load the model. See “Installing a Model Package” on page 75 for more information.

Layout

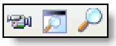


Show Grid Check this switch to show the grid in the layout window.

Show Hidden Networks Check this switch to show the invisible path networks during editing. A path network can be made to be invisible by selecting the “invisible” option for the particular network while in the Path Networks editor.

Show Routing Paths Causes routing paths to be visible during run time as well as edit time. If the option is not checked, the routing paths are visible only while in the processing editor.

View

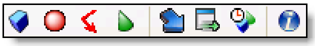


Views Allows you to define, then quickly and easily access specific areas of the model layout. Once the view is defined, you can select it while editing or running the simulation. See “Views” on page 84 for more information.

Zoom to Fit Shrinks or enlarges the layout to include the entire model.

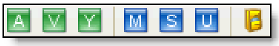
Zoom Allows you to shrink or enlarge the layout by the percentage selected.

Build Basic



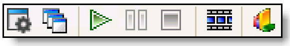
The options in this toolbar open the respective build modules. Mouse over a toolbar button to see its function and then refer to Chapter 5 for information of that build module.

Build Advanced



The options in this toolbar open the respective build modules. Mouse over a toolbar button to see its function and then refer to Chapter 6 for information of that build module.

Simulation



Simulation Options Opens the simulation options dialog. See "Simulation Options" on page 348 for more information.

Scenarios Opens the scenarios dialog. See "Scenarios" on page 353 for more information.

Play Begins the simulation

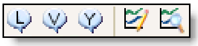
Pause Pauses and unpauses the simulation when it is running.

Stop Ends the simulation and prompts you whether to collect and view statistics up to the point where the simulation was stopped.

Animation On/Off Toggles the animation on and off

View Statistics Opens Output Viewer 3DR.

Simulation Information



Location Information Select this option and choose a location to view an information box with real time information about the location. Information for all locations may also be displayed.

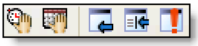
Variable Information Select this option to show the current state of all real and integer global variables.

Array Information Select this option to show the current value of all cells for arrays of up to three dimensions.

Define Dynamic Plots Allows you to define Dynamic Plots. See "Dynamic Plots" on page 364 for more information.

View Dynamic Plots Allows you to view previously defined Dynamic Plots. See "Dynamic Plots" on page 364 for more information.

Debug



User Pause by Time Allows you to specify by clock time when the simulation should pause next.

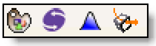
User Pause by Date Allows you to specify by calander date when the simulation should pause next. This option is not available when you are not running the simulation by calandar date.

Trace Step Select this option to step through the trace listing one event at a time. Each time you click the left mouse button, the trace will advance one event. Clicking and holding the right mouse button while in this mode generates a continuous trace.

Filtered Trace Opens the Filtered Trace dialog. See "Trace Mode" on page 361 for more information.

Debug Opens the Debug dialog. See "Debug Option" on page 357 for more information.

Tools



Graphic Editor Opens the Graphic Editor. See "Graphic Editor" on page 312 for more information.

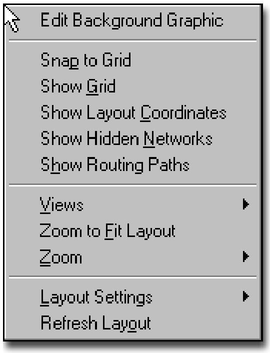
Simrunner Opens Simrunner.

Stat::Fit Opens Stat::Fit.

3D Animator Opens 3D Animator.

Right-Click Menu

To simplify many of the steps required to perform common modeling operations, ProModel includes a variety of right-click menus. From these menus, you can access context-sensitive options and settings for variables, locations, processing, path networks, resources, and other components. The following menu appears when you right-click on the layout.



The right-click layout menu includes the following options:

Edit Background Graphic Only when you have a background graphic defined, this option appears and lets you edit the graphic.

Snap to Grid Selecting this option snaps all graphics to the grid.

Show Grid When you select this option, ProModel displays the background grid.

Show Layout Coordinates Displays cursor's the coordinates in the upper-left-hand side of the layout window.

Show Hidden Networks Displays all *hidden* networks.

Show Routing Paths Displays all routing paths used in the model.

Views From this option, you may select from the views defined in the model.

Zoom to Fit Layout This option resizes the model to fit the entire image in the layout window.

Zoom Allows you to select the zoom percentage.

Layout Settings From here, you may select and define the grid settings, background color, and routing color.

Refresh Layout This option refreshes the image to reflect recent changes.

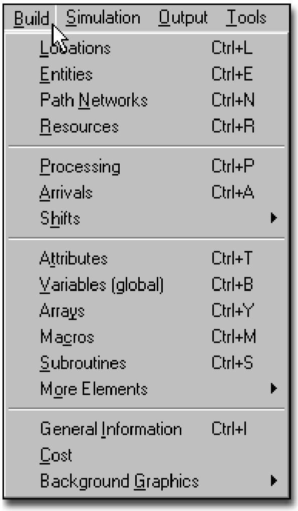
Please note

For information about the right-click menus for model elements (e.g., locations, variables, routings, resources, path networks, queues, and conveyors), see "Building the Model: Advanced Elements" on page 225.

Chapter 5: Building the Model: General Elements

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, variables, attributes, arrays, macros, and subroutines that provide the flexibility needed to model your system.



Any element represented by a graphic in the layout window can be edited by holding the CTRL key while clicking on the graphic representing that element.

How to access the Build menu:

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

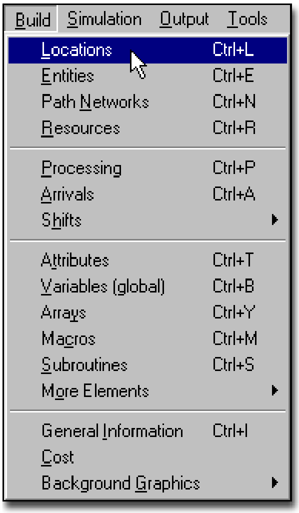
Each selection from the Build Menu is covered in detail in the following sections of this chapter.

Locations

Locations represent 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 delivery locations, warehouse locations, network servers, and transaction processing centers.

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 511.

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



How to create and edit locations

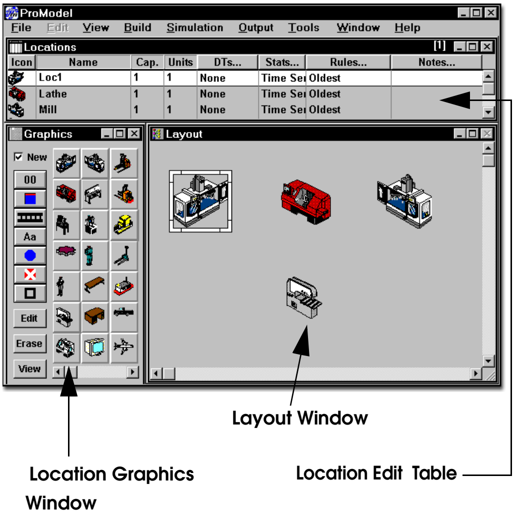
- Select **Locations** from the **Build** menu. The Locations Editor appears.

or...

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

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.

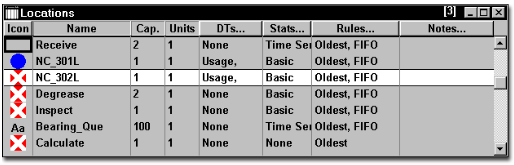


Icon	Name	Cap.	Units	DTs...	Stats...	Rules...	Notes...
	Loc1	1	1	None	Time Sei	Oldest	
	Lathe	1	1	None	Time Sei	Oldest	
	Mill	1	1	None	Time Sei	Oldest	

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.

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 Se	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 Se	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 404). 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.

Please 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 106.)

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 106.)

DTs Click on this heading button to define location downtimes, including any setup times. (See “Location Downtimes” on page 107.)

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:

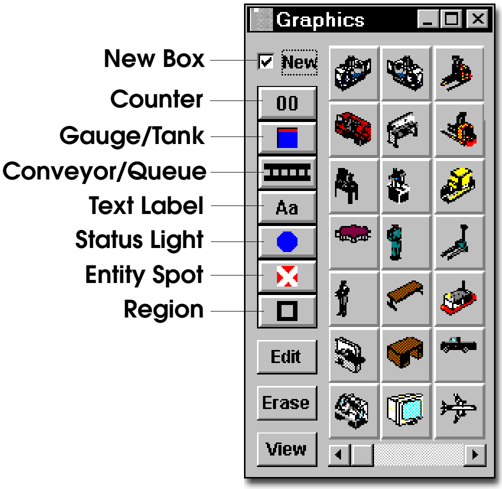
- **None** No statistics will be collected.
- **Basic** Only utilization and average time in location will be collected.
- **Time Series** Collects basic statistics and time series tracking the contents of the location over time.

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 115.) 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 button to open a larger Notes window.

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 left 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 having 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 window 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.

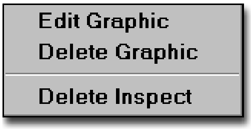
How to delete a location:

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)***.

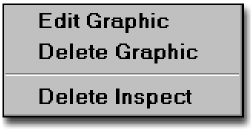
How to erase a location graphic:

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.

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 next.

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.

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 next.

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 “Text Tool” on page 324). 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 pile 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 “Naming a Graphic” on page 322).

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.

Please note

Location graphics notes:

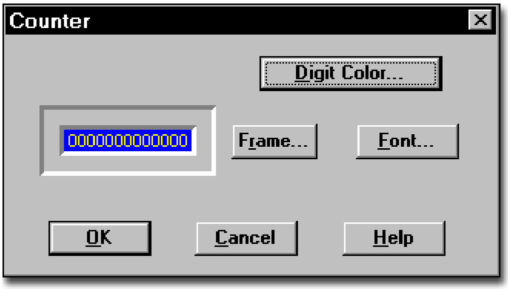
1. Location graphics are painted on the layout in the order of the location list and, for any given location having multiple graphics, in the order that the graphic was added to the layout.

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.

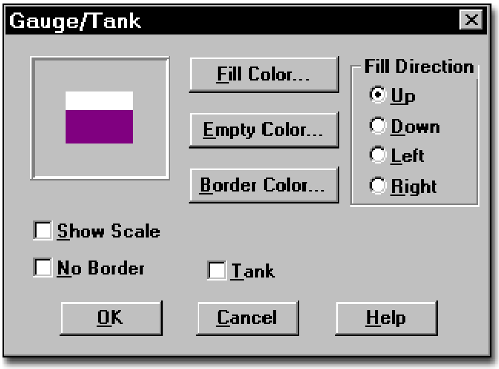
Counter Dialog Box

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.



Fill Direction: ● Up ○ Down ○ Left ○ Right

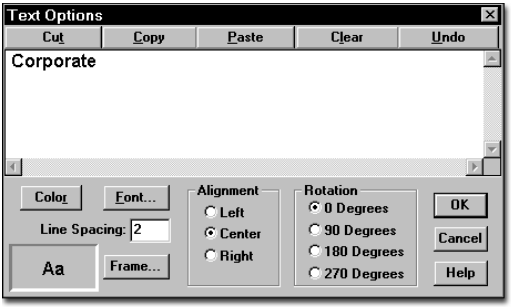
□ Show Scale

□ No Border

□ Tank

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.



Corporate

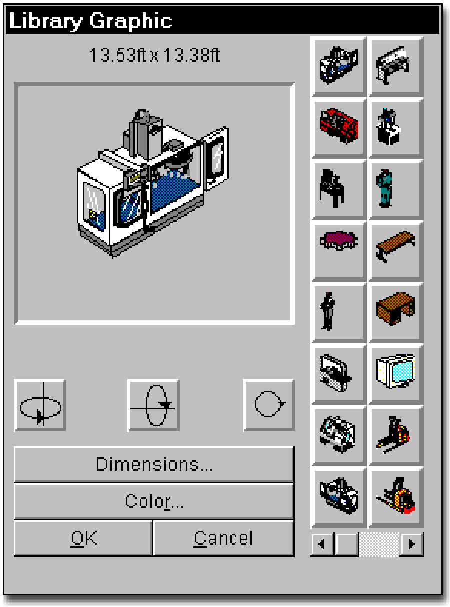
Line Spacing: 2

Alignment: ○ Left ● Center ○ Right

Rotation: ● 0 Degrees ○ 90 Degrees ○ 180 Degrees ○ 270 Degrees

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.

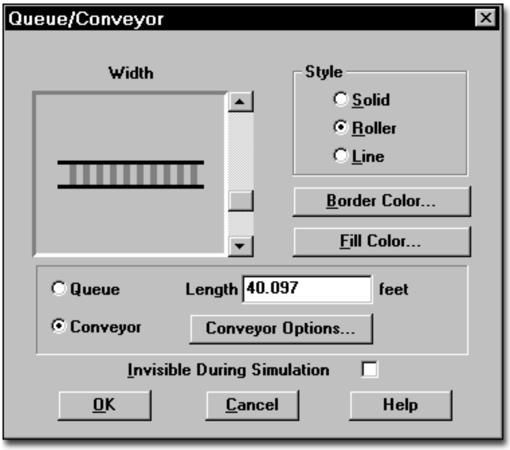


13.53ft x 13.38ft

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.

Queue/Conveyor 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 Queue/Conveyor 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 queue or conveyor. 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 during edit time and 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.



Style: ○ Solid ● Roller ○ Line

○ Queue Length 40.097 feet

● Conveyor

Invisible During Simulation □

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.

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:

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

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

Please 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. Click on the Layout window with the left mouse button where the conveyor should start.

3. Add bends to the conveyor by moving the mouse and clicking the left mouse button.

4. Move the mouse in the desired direction and click on the right mouse button to end the conveyor.

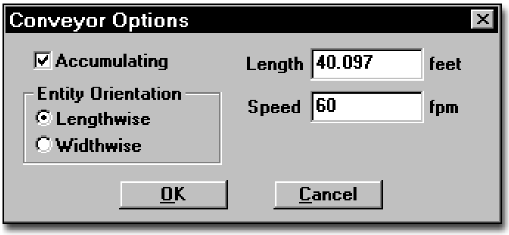
How to create bends in an existing conveyor:

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:


Conveyor Options

☑ Accumulating

Entity Orientation
● Lengthwise
○ Widthwise

Length 40.097 feet

Speed 60 fpm

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

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

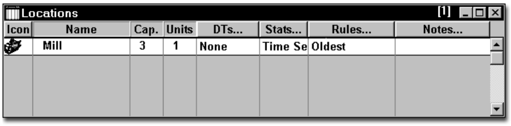
Length The length of the conveyor expressed in either feet or meters depending on the default specified in the General Information dialog.

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

Capacities and Units

Capacities

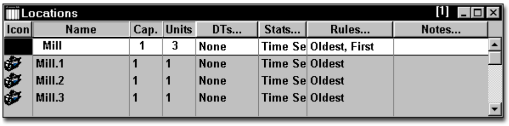
A location *capacity* is the maximum number of entities it can hold at any one time. In general, multi-capacity locations are used to model locations such as queues, waiting areas, or any other type of location where multiple entities may be held or processed concurrently. Consider the following multi-capacity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



Icon	Name	Cap.	Units	DTs...	Stats...	Rules...	Notes...
	Mill	3	1	None	Time Se	Oldest	

Units

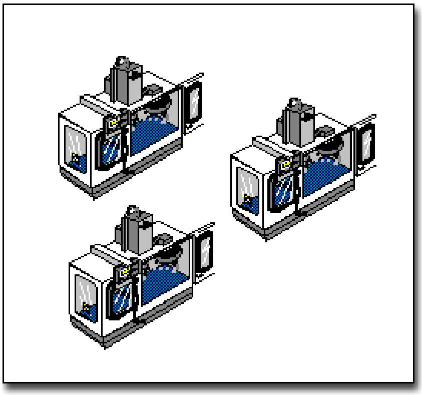
A location *unit* is defined as an independently operating station. When multiple, independently operating stations all perform the same operation and are interchangeable, they form a multi-unit location. Consider the following multi-unit location:



Icon	Name	Cap.	Units	DTs...	Stats...	Rules...	Notes...
	Mill	1	3	None	Time Se	Oldest, First	
	Mill.1	1	1	None	Time Se	Oldest	
	Mill.2	1	1	None	Time Se	Oldest	
	Mill.3	1	1	None	Time Se	Oldest	

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 workstations. Suppose, for example, we have three parallel stations, 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.

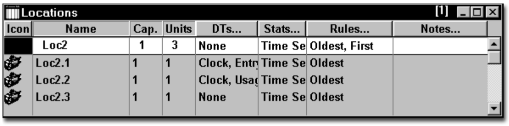
In some situations, it may even be desirable to model multi-unit locations as totally separate locations. Multiple locations should be used instead of multi-unit locations when:

- 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 the discussion regarding the Rules dialog box later in this section.)

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

Please 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.*

Location Downtimes

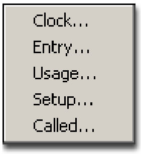
A downtime stops a location or resource from operating. A down resource or location no longer functions and is not available for use. Downtimes may represent scheduled interruptions such as shifts, breaks, or scheduled maintenance. Or, they may represent unscheduled, random interruptions such as equipment 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,a change in entity type, or called using the DOWN statement. Multi-capacity locations have only clock and called 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 for single-capactiy locaitons. (Multi-capacity locations will only have the Clock and Called options.)



3. Each selection opens an edit table for specifying the required elements of the downtime.

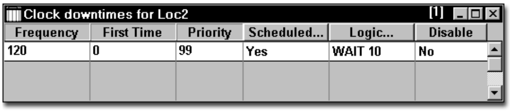
Please note

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. (See the “Appendix A” on page 587 to see if the specific statement or function is valid in a particular field.)



Frequency	First Time	Priority	Scheduled...	Logic...	Disable
120	0	99	Yes	WAIT 10	No

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, and so on.

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.

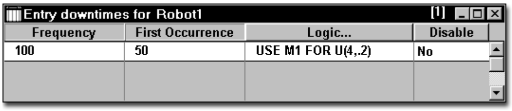
Disable Select YES to temporarily disable the downtime without deleting it from the table.

The example above 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 printer needs a new cartridge after printing 2000 shipping orders, an entity 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. The Downtime Editor contains fields for defining downtimes based on the number of entries processed at a location. Most functions, including distributions, can be included in the Frequency and First Occurrence fields. (See the "Appendix A" on page 587 to see if the specific statement or function is valid in a particular field.)



Frequency	First Occurrence	Logic...	Disable
100	50	USE M1 FOR U(4,.2)	No

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.

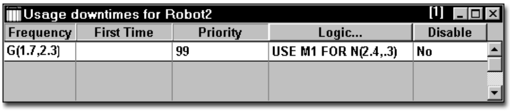
Please note

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 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 (operation time, blocked time, idle time). Usage downtimes are available only 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 the "Appendix A" on page 587 to see if a specific function is valid in a particular field.)



Usage downtimes for Robot2 [1]

Frequency	First Time	Priority	Logic...	Disable
G(1.7,2.3)		99	USE M1 FOR N(2.4,.3)	No

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.

Please note

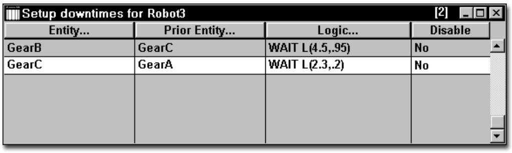
Usage-based downtimes do not accumulate. For example, if a downtime cannot occur because the priorities of the entities being processed are at least 2 levels higher than the priority of the downtime, only the first downtime resumes after processing the entities. All others are ignored.

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. 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...	Prior Entity...	Logic...	Disable
GearB	GearC	WAIT L(4.5,.95)	No
GearC	GearA	WAIT L(2.3,.2)	No

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.

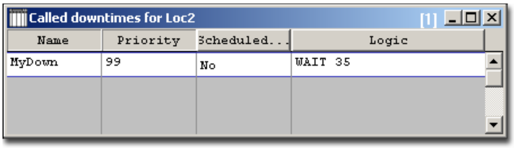
Logic Enter any logic statements to be processed when the downtime occurs. Click on the heading button to open a larger edit window.

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.

Called Downtime Editor

Called downtimes are used in conjunction with the DOWN statement to make a location go down. When the name of the called downtime is referenced during simulation by the DOWN statement, the called downtime will execute its logic.



Name	Priority	Scheduled..	Logic
MyDown	99	No	WAIT 35

Name The name of the called downtime. This is the name that will be referenced by the DOWN statement.

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, and so on.

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.

For more information on the DOWN statement, see “Down” on page 470.

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.

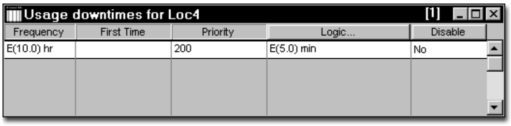
Process Table

	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 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.



Usage downtimes for Loc4 [1]

Frequency	First Time	Priority	Logic...	Disable
E(10.0) hr		200	E(5.0) min	No

The following table shows the minimum priority level requirements for an incoming entity or an 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

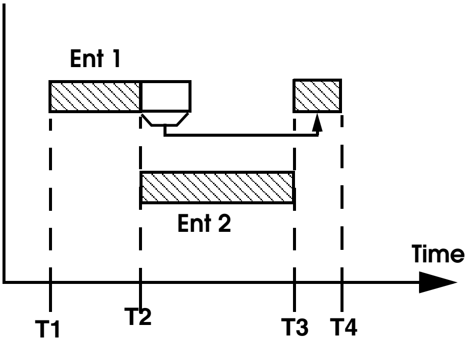
- The upper left quadrant shows that for an entity to gain access to a location already processing another entity, the incoming entity must have a priority at least one level higher than the current entity's priority.
- The upper right quadrant shows that for an incoming entity to gain access to a location 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.

Entity Preempting an Entity

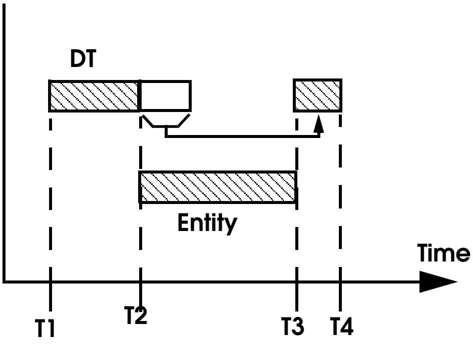


Ent 1 processing resumes upon completion of Ent 2 processing

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.

Entity Preempting a Downtime

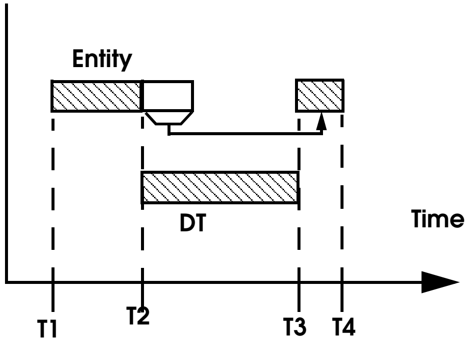


Downtime resumes immediately upon completion of entity processing

Example 3

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

Downtime Preempting an Entity

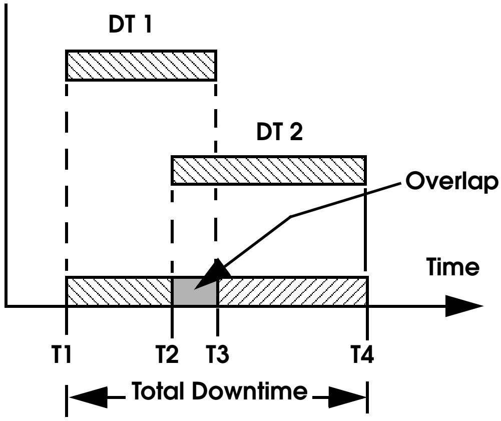


Remaining entity processing time resumes upon completion of downtime

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).

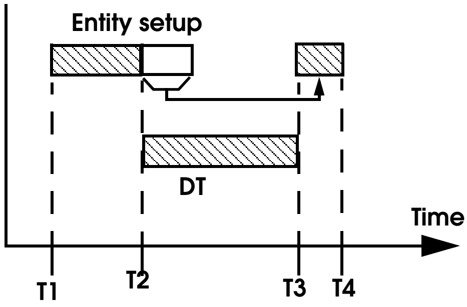
Overlapped/Concurrent Downtimes



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.

Downtime Preempting Entity in Setup

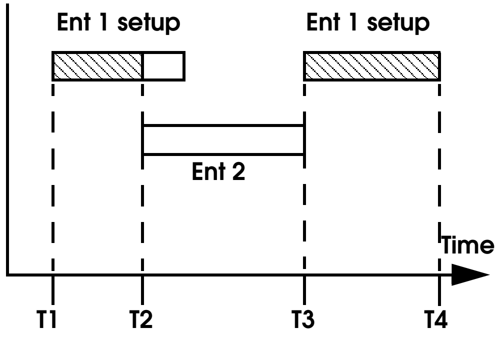


Remaining setup time resumes upon completion of downtime

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.

Entity Preempting Entity in Setup



Ent 1 setup must start over upon completion of Ent 2 processing

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 downtime inhibiting statement.

Downtime inhibiting statements

WAIT UNTIL

ACCUM

COMBINE

MATCH

GROUP

JOIN

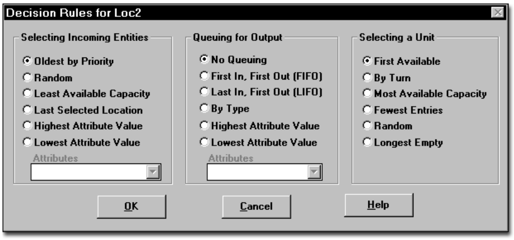
LOAD

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 300.

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



Decision Rules for Loc2

Selecting Incoming Entities
● Oldest by Priority
○ Random
○ Least Available Capacity
○ Last Selected Location
○ Highest Attribute Value
○ Lowest Attribute Value
Attributes

Queuing for Output
● No Queuing
○ First In, First Out (FIFO)
○ Last In, First Out (LIFO)
○ By Type
○ Highest Attribute Value
○ Lowest Attribute Value
Attributes

Selecting a Unit
● First Available
○ By Turn
○ Most Available Capacity
○ Fewest Entries
○ Random
○ Longest Empty

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 that has been waiting the longest. Location attributes are also valid entries.

Lowest Attribute Value Selects the entity which has 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 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.

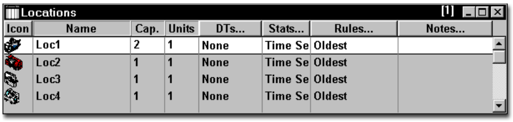
Random Selects an available unit randomly.

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.

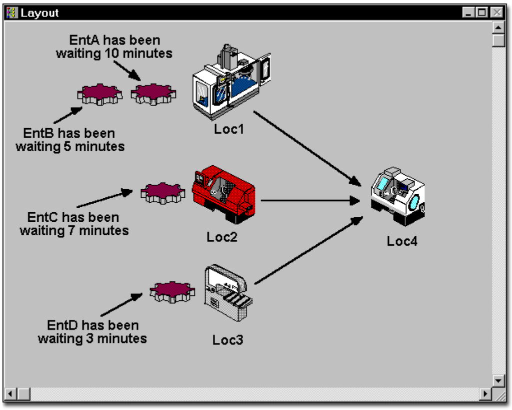
Rules Dialog Box Example

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	



Layout
EntA has been waiting 10 minutes
EntB has been waiting 5 minutes
Loc1
EntC has been waiting 7 minutes
Loc2
Loc4
EntD has been waiting 3 minutes
Loc3

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.

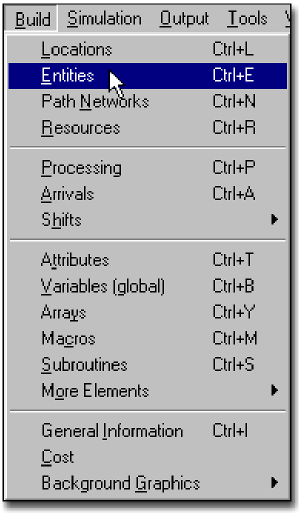
The part to enter Loc4 first is EntC at Loc2 which waited 7 minutes. Even though EntA has been waiting ten minutes, it must wait until EntB has been routed, because EntB is ahead of it in the output queue according to the LIFO queuing rule. Once Loc4 finishes processing EntC, EntB at Loc1 enters Loc4.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

Entities

Anything that a model processes is called an “Entity.” Documents, people, or phone calls 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 two or more documents are joined together (through the JOIN statement); split into two or more entities, such as when a box is opened and the contents removed (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 476 for more information.

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



How to access the entities editor:

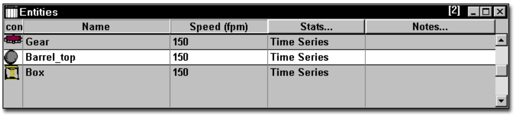
- Select **Entities** from the **Build** menu.

or...

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

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.



Entities

con	Name	Speed (fpm)	Stats...	Notes...
	Gear	150	Time Series	
	Barrel_top	150	Time Series	
	Box	150	Time Series	

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 120.

Name The entity name. See “Names” on page 404 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.

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.

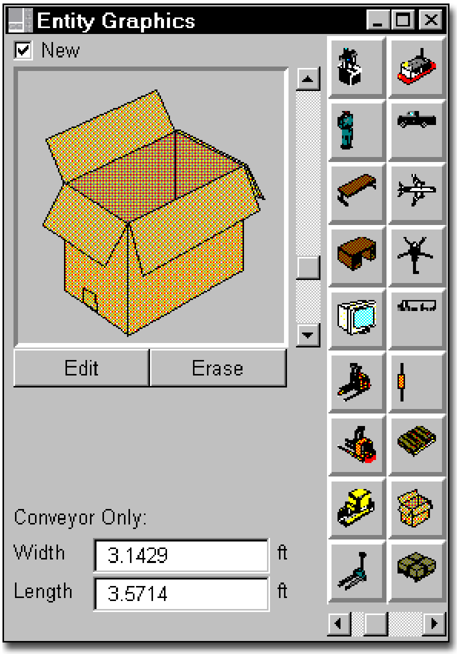
How to define entities graphically:

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.



☑ New

Conveyor Only:

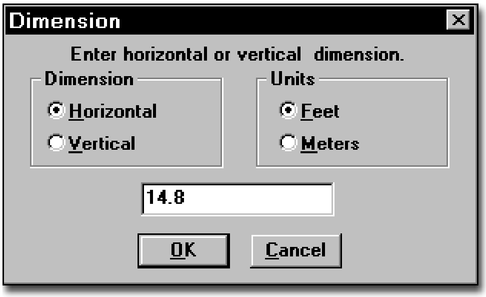
Width 3.1429 ft

Length 3.5714 ft

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.


Dimension

Enter horizontal or vertical dimension.

Dimension: ● Horizontal ○ Vertical

Units: ● Feet ○ Meters

14.8

Please note

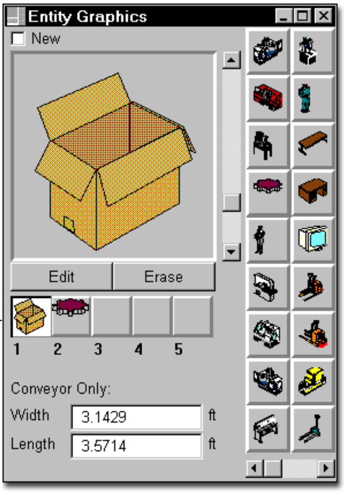
Since the horizontal and vertical dimensions must remain proportional, only one of the dimensions needs to be changed. The other dimension changes automatically to maintain proportionality.

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 loan application could be assigned three graphics: the first representing the application before a credit check has been received, the second representing the application after receiving the credit check but before the loan is approved, and the third representing the application after the loan is approved. During the simulation, the application’s status could be shown by adding additional graphics to represent each state of the loan process using the GRAPHIC statement (see “Graphic” on page 492 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.

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 111).

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 107 and “Resource Downtimes” on page 138 for more details on entity preemption.)

Process Table

	Location	Operation (min)
EntA	Loc1	Use Res1,99 For N(3,.1)

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2,200	First 1	MOVE FOR 2.5

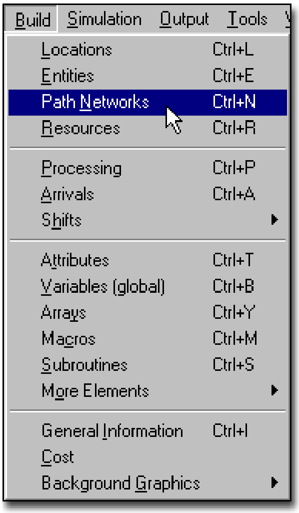
Path Networks

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 path networks are described in more detail in the section “Crane Systems” on page 269.

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 uni-directional or bi-directional. Multiple path segments, which may be straight or jointed, may be connected at path nodes. 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, which is accessed from the Build menu.



How to create or edit a path network:

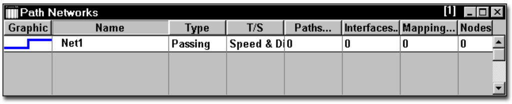
- Select **Path Networks** from the **Build** menu.

or...

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

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 or Passing), 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.



Graphic	Name	Type	T/S	Paths...	Interfaces..	Mapping...	Nodes
	Net1	Passing	Speed & Di	0	0	0	0

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. 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 404.

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.

A "Crane" option is also available, which is described in more detail in the section “Crane Systems” on page 269.

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.

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.

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.)

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 132). The Nodes edit table is covered in more detail later in this section.

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** or **Non-passing** as the network type.

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.

Please 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 both on the layout and in the Nodes edit table.

3. Drag an existing node to move that node.

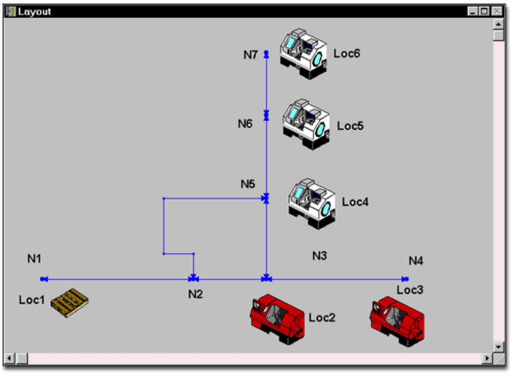
How to move a path network:

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.

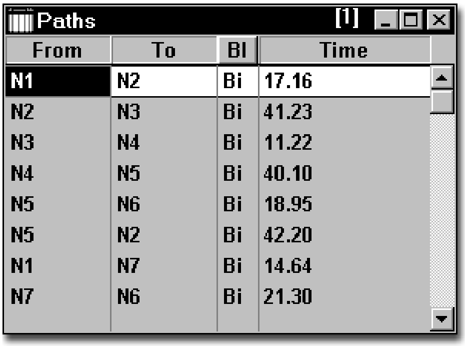
A Typical Path Network

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.



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."



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.

Please 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 149 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 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.

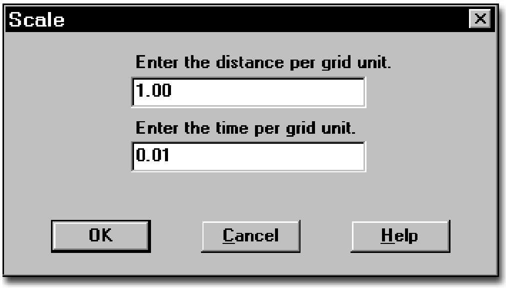
How to set the default time and distance values per grid unit:

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.



Scale

Enter the distance per grid unit.
1.00

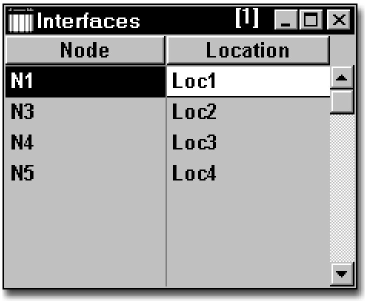
Enter the time per grid unit.
0.01

Please note

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

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.



Node	Location
N1	Loc1
N3	Loc2
N4	Loc3
N5	Loc4

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.

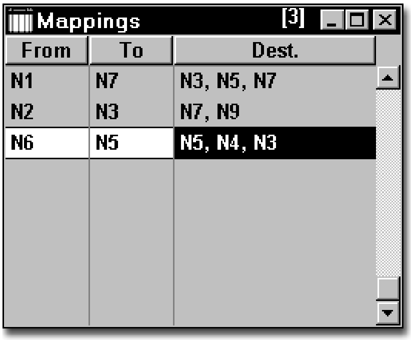
2. Left click on the desired location to complete the interface.

Please 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 interface only with the "parent" location (i.e., Loc1) of a multi-unit location.

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.



From	To	Dest.
N1	N7	N3, N5, N7
N2	N3	N7, N9
N6	N5	N5, N4, N3

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.

Please note

When your simulation is compiled and run, ProModel will automatically define destinations for your network mapping, if you have not defined them yourself. These computer-generated mappings will not appear in the Dest. column of the Mappings table, but the From and To columns will contain information on these mappings. Do not delete the information in these collumns.

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.

Please 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

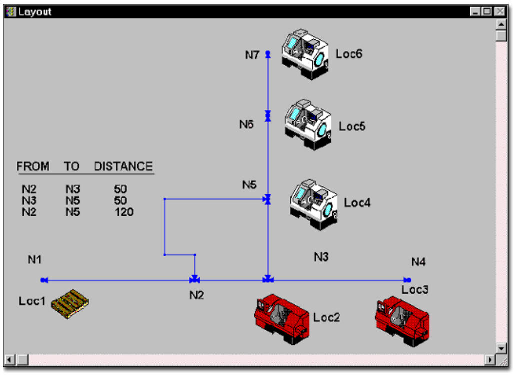
The following example uses a path network diagram to demonstrate mapping.

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.

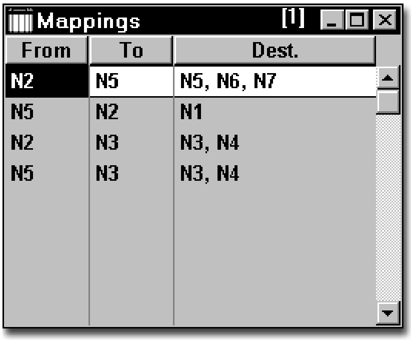
Because the combined length of segments connecting N2 to N3 and N3 to N5 is shorter than the length of the single segment from N2 to N5, resources and entities based on speed and distance will normally take the former path to travel.

To force them to take the longer path, we must specify mapping constraints.



FROM	TO	DISTANCE
N2	N3	50
N3	N5	50
N2	N5	120

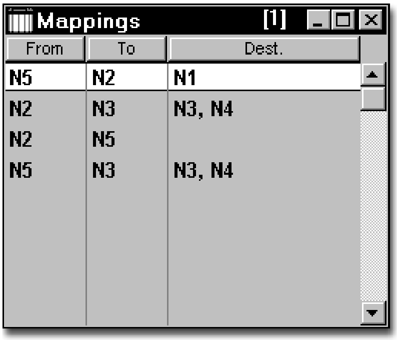
This case requires two explicit mapping constraints to override the selection of default paths in each direction: The first table entry forces entities and resources en-route from Loc1 to Loc4, Loc5, or Loc6 to override the default path and take the direct branch from N2 to N5. The second table entry 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.



From	To	Dest.
N2	N5	N5, N6, N7
N5	N2	N1
N2	N3	N3, N4
N5	N3	N3, N4

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, ..., and locations Loc7, Loc8, ..., 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>".



From	To	Dest.
N5	N2	N1
N2	N3	N3, N4
N2	N5	
N5	N3	N3, N4

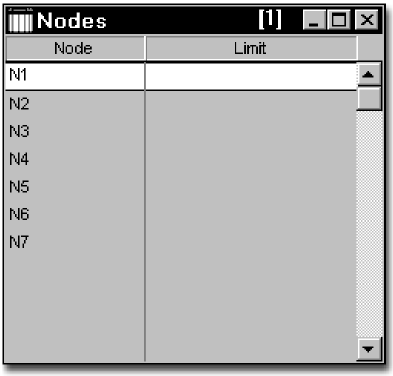
The empty destination column will be interpreted as "all other destination nodes" by ProModel.

Please 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.

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 132).



Node	Limit
N1	
N2	
N3	
N4	
N5	
N6	
N7	

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.

Pre-translation check for Path Networks

When you run your simulation, ProModel will compile your model and check your path networks for errors. If your model has large, complex path networks, this could cause your model’s compilation time to run quite long.

To combat this, ProModel will perform a pre-translation check for path networks. This means that the first time your model compiles, it will check your path networks for errors. It will then save this information.

Every subsequent time your model compiles, it will read the saved information about your path networks and not check the path networks for errors. This saves compilation time on large models.

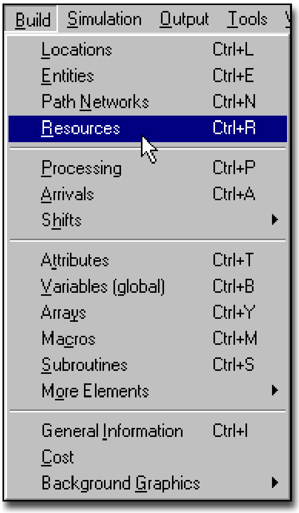
However, if you edit one or more of your path networks, ProModel will once again check the modified path networks for errors during compilation. ProModel will consider a path network to have been modified if its name was at any time, since the last compile, highlighted in the Path Network dialog window.

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 with 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. 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 "Name-Index Numbers" on page 406 and "Res()" on page 543 for more information.

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



How to create and edit resources:

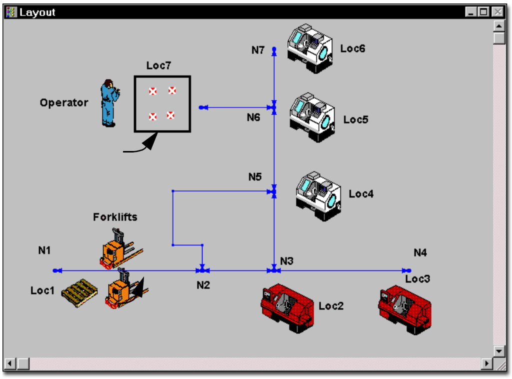
- Select **Resources**... from the **Build** menu.

or...

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

Typical Use of Resources

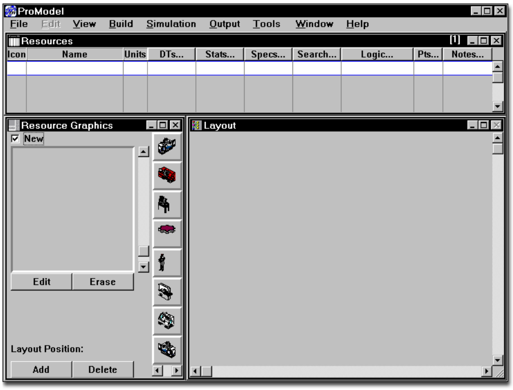
The diagram below shows two types of resources: forklifts and an operator. Forklifts are used as resources to transport entities from Loc1 to any of the processing locations, Loc2 through Loc6. The forklifts are dynamic resources and travel along the path network, Net1, explained previously in "Path Networks" on page 123. 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.

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.

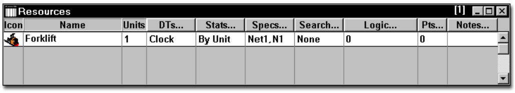

ProModel
File Edit View Build Simulation Output Tools Window Help
Resources
Icon Name Units DTs... Stats... Specs... Search... Logic... Pts... Notes...
Resource Graphics
☑ New
Edit Erase
Layout Position:
Add Delete
Layout

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 for movement.

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, or its status. 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.


Resources

Icon	Name	Units	DTs...	Stats...	Specs...	Search...	Logic...	Pts...	Notes...
	Forklift	1	Clock	By Unit	Net1,N1	None	0	0	

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). 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 also “Resource Downtimes” on page 138.)

Please 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.
- **Summary:** 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. From here you can assign a path network, set the resource speed, and define pickup and deposit times. For more information on the Specification dialog, see “Resource Specifications Dialog Box” on page 143.

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. Resource points are auxiliary points where multiple resources may appear graphically when parked or in use at a multi-capacity node.

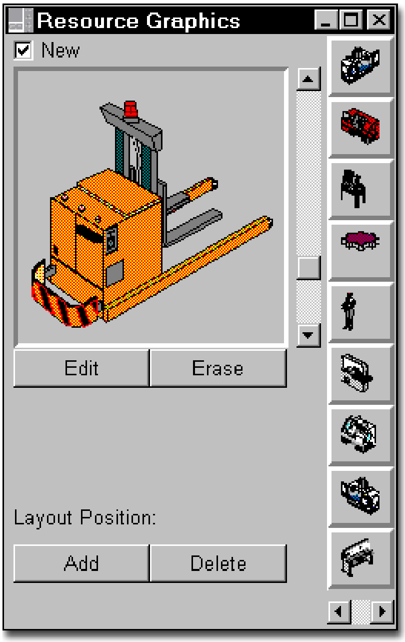
Notes... Enter any notes in this field, or click on the heading button to open a larger Notes window for entering notes.

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 specify the dimensions of the resource graphic. For more information, see “Dimension” on page 318.



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.

Static Resources

Static resources are resources 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.

Please 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.*

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, which are 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 147 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.

Please 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.

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 492) 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.

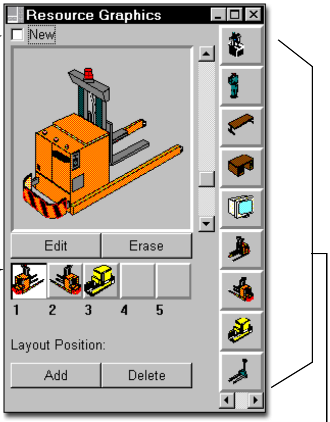
6. Change the color, rotation, or other aspect of the new graphic by clicking on the Edit button.

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

Multiple Resource Graphics Example

Uncheck the New Box

Select the next blank resource graphic cell



Select an icon for the resource

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

Please 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 above. 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 492 for more information.

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, Technician, which has ten units. You also have ten locations and each resource unit can only interface with one location. For example, Technician.5 can only work at Loc5. Since "USE Technician.5" is invalid, you would need to use multiple single-unit resources instead (e.g., Technician1, Technician2, Technician3).

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 computer can be operated by one of three technicians. If you did not use multi-unit resources, you would need to specify "USE Technician1 OR Technician2 OR Technician3," 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.

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 168). 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**.

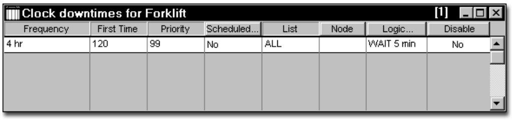
5. Enter the required information in either the Clock Downtime or Usage Downtime edit table. Each of these tables is described in the following pages.

Please 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:



Clock downtimes for Forklift [1]

Frequency	First Time	Priority	Scheduled...	List	Node	Logic...	Disable
4 hr	120	99	No	ALL		WAIT 5 min	No

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.

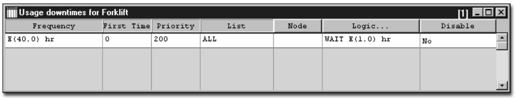
Node This field applies only to dynamic resources and defines the node to which the resource will travel to go down. If no node is entered, the resource stays at the current node. The actual downtime will not begin until the resource arrives at this node. Traveling to the downtime node is counted statistically 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 the *"Appendix A" on page 587* 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	First Time	Priority	List	Node	Logic...	Disable
E(40.0) hr	0	200	ALL		WAIT E(1.0) hr	No

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 the “Appendix A” on page 587).

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

Please 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.

Resource Priorities and Preemption

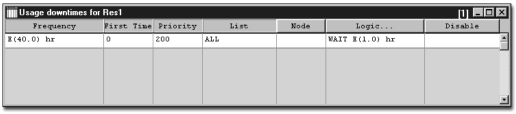
Priorities for resource requests may be assigned through a GET, JOINTLY GET, or USE statement in operation logic, downtime logic, or move logic (or the subroutines called from these logics). Priorities for resource downtimes are assigned in the Priority field of the Clock and Usage downtime edit tables.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

Process Table

	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



Frequency	First Time	Priority	List	Node	Logic...	Disable
E(40.0) hr	0	200	ALL		WAIT E(1.0) hr	

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.

Please 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.

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 still indicate the additional time spent working on the downed machine.

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.

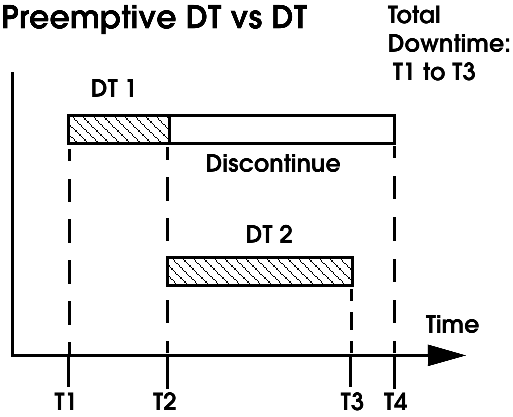
Priority values are divided into ten levels (0 to 99, 100 to 199,..., 900-999), with values beyond 99 used for preempting entities or downtimes of a lower priority level.

	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 or a location’s 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 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.

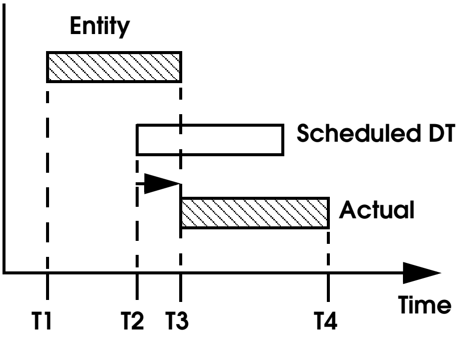
The following graphics demonstrate basic preemption concepts.

Preemptive DT vs DT



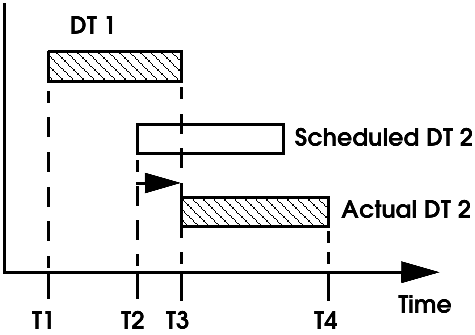
DT 2 priority at least 1 level higher than DT 1 priority

Non-preemptive DT vs. Entity



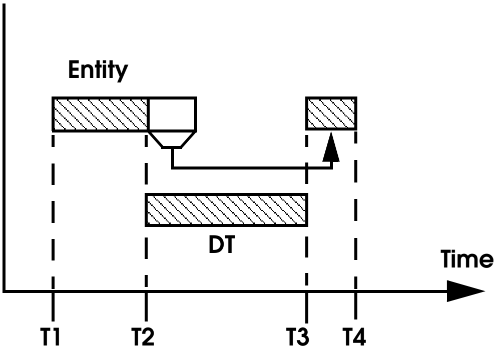
DT 2 priority NOT at least 1 level higher than entity 1 priority

Non-preemptive DT vs. DT



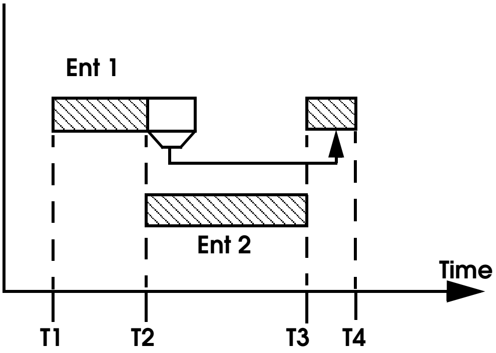
DT 2 priority NOT at least 1 level higher than DT 1 priority

Preemptive DT vs. Entity



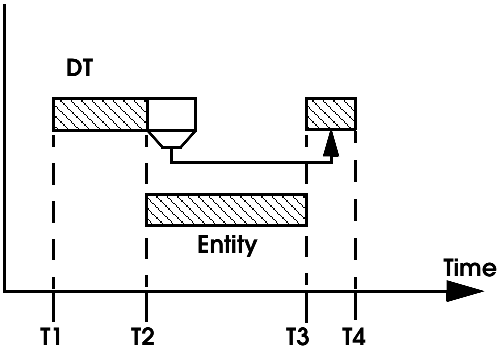
DT priority at least 1 level higher than entity priority

Preemptive Entity vs. Entity



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

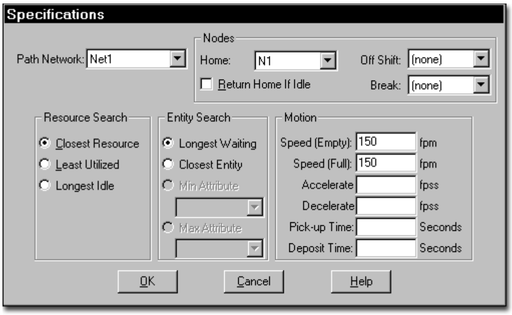
Preemptive Entity vs. DT



Entity priority at least 2 levels higher than DT priority

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 with path networks). If the resource is static (i.e., not assigned to a path network) many of the options will be disabled. The Resource Specifications dialog box includes the following fields:


Specifications

Path Network: Net1

Nodes
Home: N1
□ Return Home If Idle
Off Shift: (none)
Break: (none)

Resource Search
● Closest Resource
○ Least Utilized
○ Longest Idle

Entity Search
● Longest Waiting
○ Closest Entity
○ Min Attribute
○ Max Attribute

Motion
Speed (Empty): 150 fpm
Speed (Full): 150 fpm
Accelerate fpss
Decelerate fpss
Pick-up Time: Seconds
Deposit Time: Seconds

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 where the resource is positioned at the beginning of the simulation.

Please 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**

Please 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**

Please 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 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**

Please 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.

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).

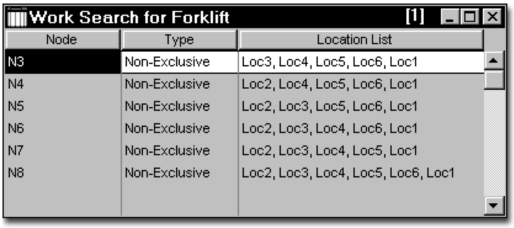
Park Search Park searches are typically used to get a resource off a main path segment, or send the resource to the next most likely place work will become available. A park search is a list of *nodes* to which a resource may be sent to park if no work is waiting at either the work or default search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

Please 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).



Node	Type	Location List
N3	Non-Exclusive	Loc3, Loc4, Loc5, Loc6, Loc1
N4	Non-Exclusive	Loc2, Loc4, Loc5, Loc6, Loc1
N5	Non-Exclusive	Loc2, Loc3, Loc5, Loc6, Loc1
N6	Non-Exclusive	Loc2, Loc3, Loc4, Loc6, Loc1
N7	Non-Exclusive	Loc2, Loc3, Loc4, Loc5, Loc1
N8	Non-Exclusive	Loc2, Loc3, Loc4, Loc5, Loc6, Loc1

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

Node The node for which the work search is defined.

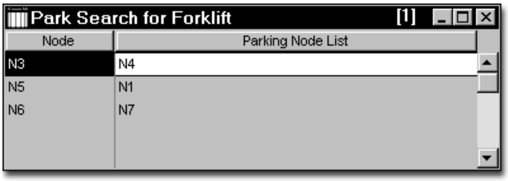
Type Exclusive or non-exclusive work search. See “Resource Search Routines” on page 145 for an explanation of *exclusive* and *non-exclusive* work searches.

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).



Park Search for Forklift [1]

Node	Parking Node List
N3	N4
N5	N1
N6	N7

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

Node The node for which the park search is defined.

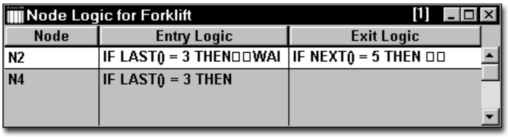
Parking Node List A list of *nodes* to which a resource may be sent to park. The resource will look for available capacity at the first node in the list (Node capacity is defined in the Path Networks edit table in the Nodes window). If there is no capacity at that node, the resource will look to the 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.

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



Node Logic for Forklift [1]

Node	Entry Logic	Exit Logic
N2	IF LAST() = 3 THEN□□WAI	IF NEXT() = 5 THEN □□
N4	IF LAST() = 3 THEN	

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

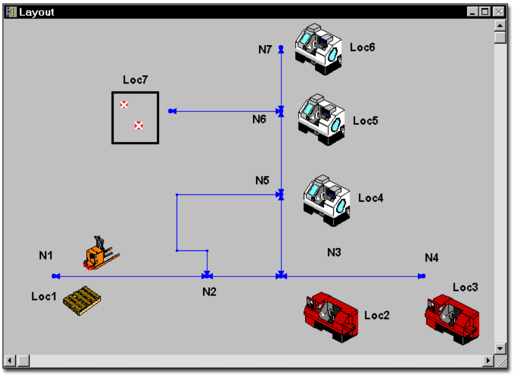
Node The entry or departure node where the resource will process the logic.

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.

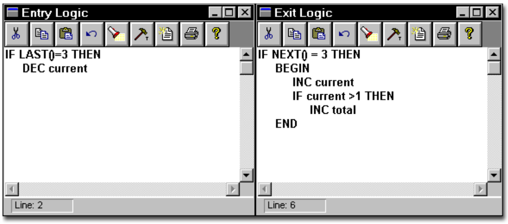
Node Logic Example



Suppose that for safety considerations we desire to keep track of the number of times 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.



```
IF LAST()=3 THEN
    DEC current
```

```
IF NEXT() = 3 THEN
    BEGIN
        INC current
        IF current >1 THEN
            INC total
    END
```

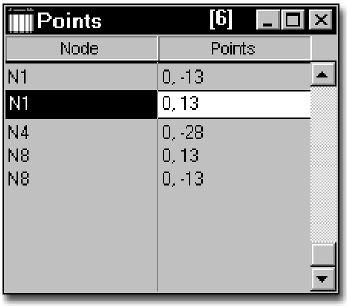
Please note

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 "Resource-Specific System Functions" on page 435).

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 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 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 N8 has two resource points attached to it. The horizontal offset is 0 units for each point. The vertical offset is 13 units both up and down from the node position. (For resource points positive distances are up and to the right.)



Node	Points
N1	0, -13
N1	0, 13
N4	0, -28
N8	0, 13
N8	0, -13

Whenever a forklift arrives or parks at node N8, 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 N8 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.

Please note

Resource points are automatically added to the home node for each resource graphic placed on the layout.

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:

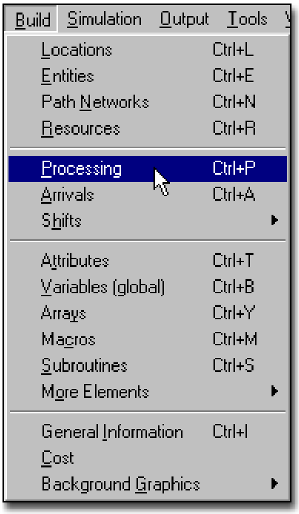
1. Bring up the Resource Points edit table by clicking on the **Pts...** heading button from the Resources edit table.

2. Drag the resource point to a new location with the mouse.

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**.

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.

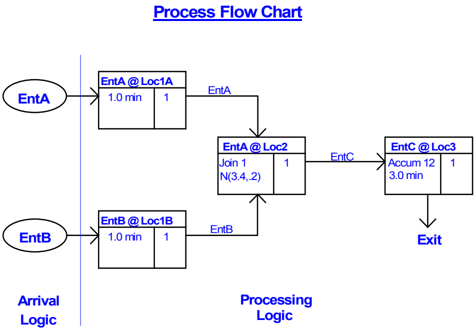
Defining processes graphically in ProModel requires interaction with all four process editing windows.

- 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.



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.

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.

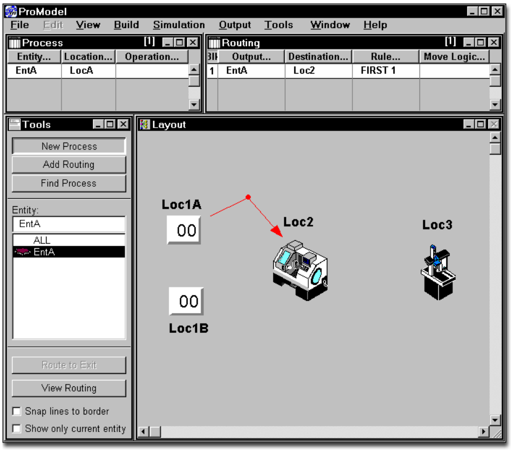
3. Click on the first location where the entity will process. A rubber-banding routing line appears. If you select Add Routing, the rubber-banding routing arrow automatically appears from the current location.

4. To choose a different entity as the output entity, select the desired output entity in the tools window.

5. Click the destination location.

Please note

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.



Entity...	Location...	Operation...
EntA	LocA	

Blk	Output...	Destination...	Rule...	Move Logic...
1	EntA	Loc2	FIRST 1	

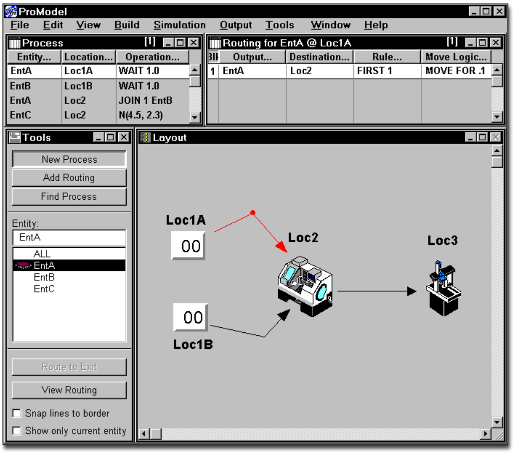
Entity: EntA

□ Snap lines to border
□ Show only current entity

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.



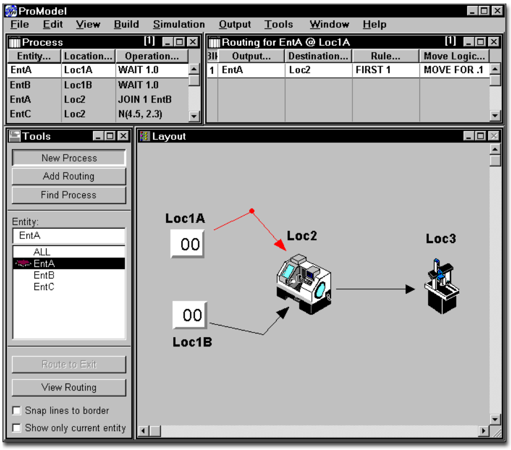
Entity...	Location...	Operation...
EntA	Loc1A	WAIT 1.0
EntB	Loc1B	WAIT 1.0
EntA	Loc2	JOIN 1 EntB
EntC	Loc2	N(4.5, 2.3)

Bl	Output...	Destination...	Rule...	Move Logic...
1	EntA	Loc2	FIRST 1	MOVE FOR .1

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 Logic Builder, documented at the end of this section.

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.



Entity...	Location...	Operation...
EntA	Loc1A	WAIT 1.0
EntB	Loc1B	WAIT 1.0
EntA	Loc2	JOIN 1 EntB
EntC	Loc2	N(4.5, 2.3)

Bl	Output...	Destination...	Rule...	Move Logic...
1	EntA	Loc2	FIRST 1	MOVE FOR .1

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

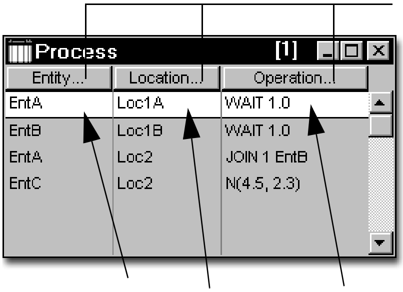
Routing Edit Table Appears in the upper right corner of the workspace and controls the destination of entities that have finished at the location.

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.

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.



Entity...	Location...	Operation...
EntA	Loc1A	WAIT 1.0
EntB	Loc1B	WAIT 1.0
EntA	Loc2	JOIN 1 EntB
EntC	Loc2	N(4.5, 2.3)

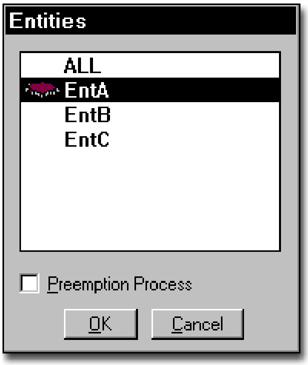
Heading Buttons

Current Entity, Location, and Operation are highlighted.

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 300) and 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.



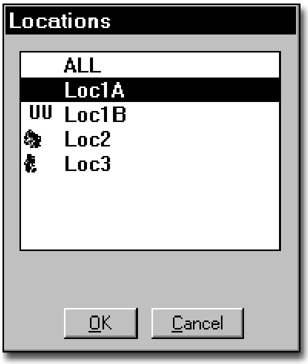
ALL
EntA
EntB
EntC

□ Preemption Process

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.



ALL
Loc1A
Loc1B
Loc2
Loc3

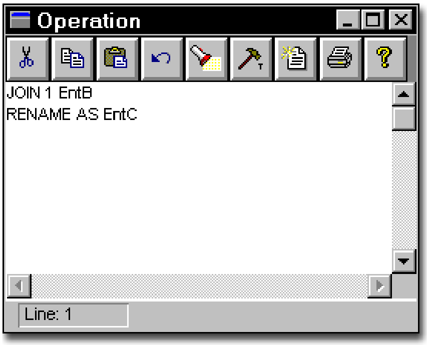
The location list box defaults to the current location, the last location selected, or the first location defined.

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 “Statements and Functions” on page 439.

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.



```
JOIN 1 EntB
RENAME AS EntC
```

For more information see “Operation Logic” on page 299.

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 packing station for packaging. The packing time is .5 minutes and the entities move on to a shipping 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

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

Routing Table

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

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

Routing Table

	Output	Destination	Rule	Move Logic
1	EntA	Ship1	FIRST 1	MOVE FOR 1
1	EntB	Ship2	FIRST 1	MOVE FOR 1
1	EntC	Ship3	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	Packaging	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

	Location	Operation (min)
EntA	Loc1	WAIT .4
EntB	Loc1	WAIT .5
EntC	Loc1	WAIT .6
ALL	Loc1	

Routing Table

	Output	Destination	Rule	Move Logic
1	**ALL**	Packaging	FIRST 1	MOVE FOR 1

Alternatively, you can assign an attribute to each entity which represents 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

Entity	Location	Operation (min)
ALL	Test	WAIT Oper_Time

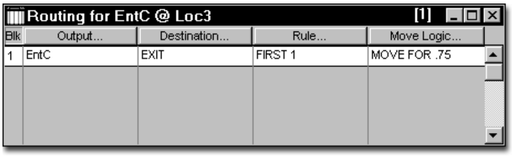
Routing Table

Blk	Output	Destination	Rule	Move Logic
1	**ALL**	Packaging	FIRST 1	MOVE FOR 1

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 subtable 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 541).


Routing for EntC @ Loc3

Blk	Output...	Destination...	Rule...	Move Logic...
1	EntC	EXIT	FIRST 1	MOVE FOR .75

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 552). 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

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

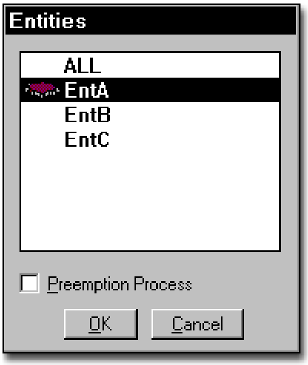
Routing Table

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, or it may be 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 likely to be made, the entity highlighted in the list box defaults first to the current field entity, then to the last entity selected, 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

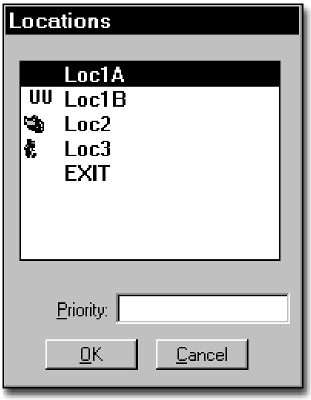
	Location	Operation (min)
EntA	Loc1	WAIT T(2,5,8)

Routing Table

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

Destination This is the location to which entities are to 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:



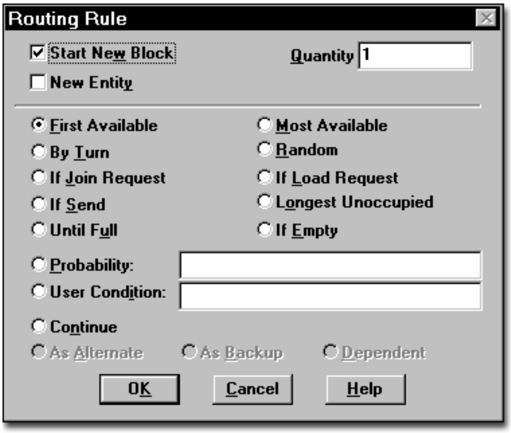
Instead of entering a location name, you may use the LOC() function which returns the location name corresponding to the index designated by an expression. For example, LOC(5) routes to the fifth location defined or the fifth record in the location edit table. You can also specify variable destinations 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 511 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 like 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 159.)

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.



Routing Rule

☑ Start New Block   Quantity 1

□ New Entity

● First Available   ○ Most Available
○ By Turn   ○ Random
○ If Join Request   ○ If Load Request
○ If Send   ○ Longest Unoccupied
○ Until Full   ○ If Empty
○ Probability:
○ User Condition:
○ Continue
○ As Alternate   ○ As Backup   ○ Dependent

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.

Please 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 mail_bag enters a location and is separated into 30 individual pieces of mail, called Letters. Enter 30 as the quantity and Letter as the output entity, as in the example below. Only the first line in a routing block can specify a quantity.

Process Table

	Location	Operation (min)
mail_bag	Loc1	WAIT T(2,5,8)

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Letter	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 415.

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.

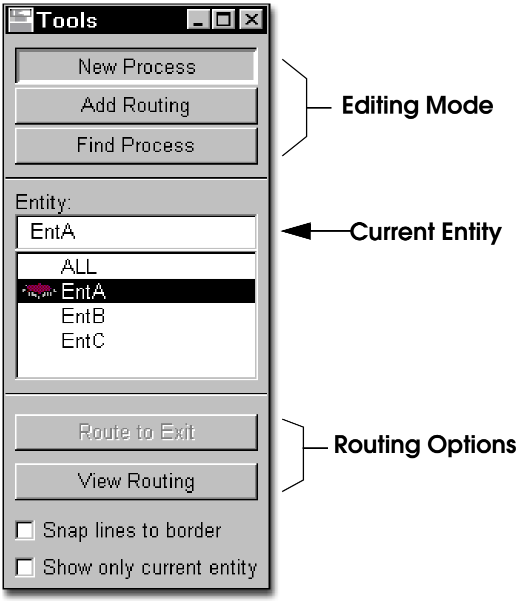
When defining exit logic, such as incrementing a variable used to track the number of exits from a location, it can generally go before the move statement unless a MOVE WITH statement is used and the entity must capture the resource before making the move. In this situation, a GET statement should be 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 on “Routing Move Logic” on page 159.

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.

Please 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

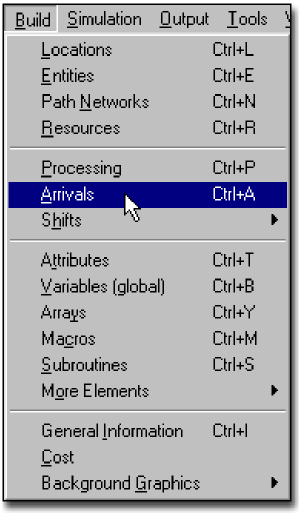
When you highlight an entity in the list box, checking this option will show only those routings associated with the entity.

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.

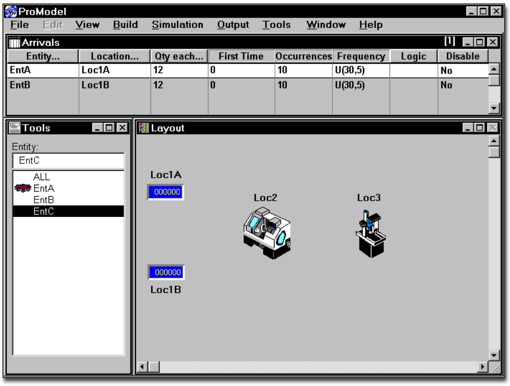


How to edit arrivals

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

Arrivals Editor

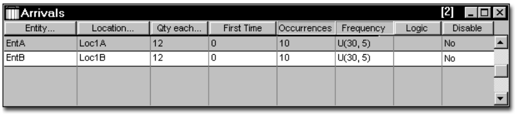
The Arrivals Editor consists of three windows that appear on the screen together. The Arrivals edit table contains the specifications of each arrival to the system and appears across the top of the screen. The Tools window contains tools for defining arrivals graphically and appears at the bottom left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



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

Arrivals Edit Table

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



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.

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.

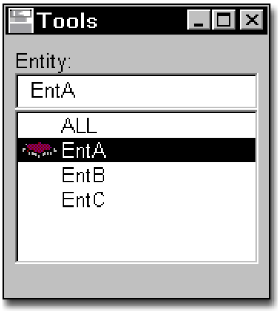
Defining Arrivals

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 and frequency).

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 using either the keyboard or the statement builder (accessed by right clicking inside the desired field).

4. Enter the time of the first arrival by clicking on the **First Time** button. (See "Independent Arrivals" on page 165 for more information on the First Time dialog.)

5. Click on the **Logic** or **Notes** heading button to enter desired logic or notes.

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 appointments, meeting times, or pickup and delivery times. When defining independent arrivals, remember that simulation can model only *predefined* appointment schedules. This means that dynamically scheduled appointments (e.g., rescheduling return visits to fit into available slots) must take place where you define the appointment 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., interviews scheduled every fifteen minutes).

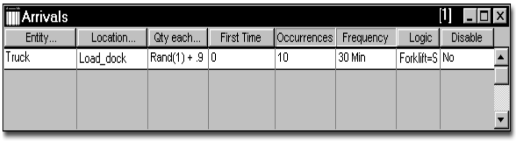
Please note

When you define independent arrivals as intervals, the next arrival time is independent of the previous arrival. For example, applicants will arrive for appointments 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., applicants must arrive at least ten minutes prior to their interviews).
- Define a distribution to allow variability from the adjusted arrival time.
- Allow the possibility that an entity will not arrive at all.
- Define specific appointment types for only certain resources or resource types.

How to define independent arrivals:

1. Open the **Arrivals** module from the **Build** menu.



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

2. Click on the **Entity** button and select the entity type you want to schedule.

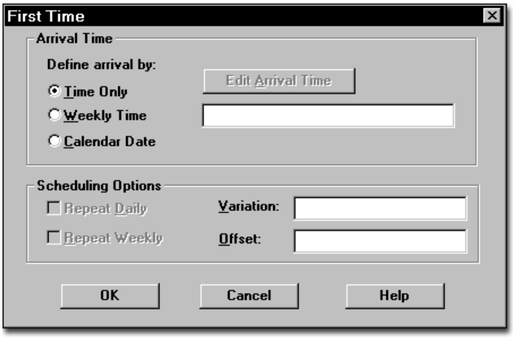
3. Click on the **Location** button and select the location where you want the independent arrival to appear.

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 applicants show up for scheduled interviews, 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 appointments occurring at equal intervals, this is the time of the first appointment 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.)

Please note

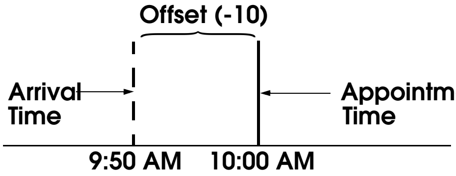
The arrival time must match the time units selected for the simulation run-length. If you defined the elapsed time by calendar date, you must also define the simulation length by calendar date. To edit arrival times, select either day and time or calendar date and click the Edit Arrival Time button.



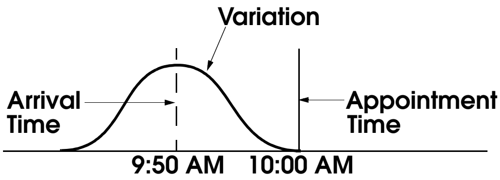
Define arrival by:
● Time Only
○ Weekly Time
○ Calendar Date

□ Repeat Daily
□ Repeat Weekly
Variation:
Offset:

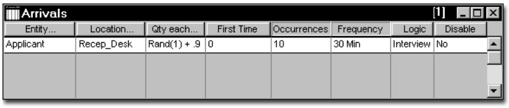
6. In the offset field, enter any offset you wish to apply to the arrival time. This will direct incoming appointments to arrive earlier (or later) than scheduled. For example, if job applicants are to arrive 10 minutes prior to the time of their appointment, enter "-10"



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 appointment definition. (Enter 1 if it will occur only once.)



Entity...	Location...	Qty each...	First Time	Occurrences	Frequency	Logic	Disable
Applicant	Recep_Desk	Rand(1) + .9	0	10	30 Min	Interview	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 First Arrival dialog applies it to each occurrence.

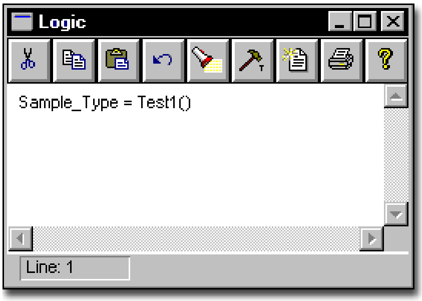
10. In the Logic field of the arrivals edit table, enter any attribute assignments you will use to determine the processing of the entity. This might include the resource or appointment type (e.g., interview) and it will be helpful if you define these attributes beforehand.

Arrival Logic

Arrival Logic allows you to perform certain logic as an entity enters the system and is used primarily for assigning initial entity attribute values. Suppose you process three different types of samples at an inspection station and each sample takes either 8, 10, or 12 minutes to test. Fifty percent of the samples take 8 minutes to test, 35% take 10 minutes, and 15% take 12 minutes.

The samples have different processing times depending on the test performed. To differentiate between the different types of samples, we assign an entity attribute called Sample_Type to the samples. We define a discrete, non-cumulative user distribution called Test1 with the following information:

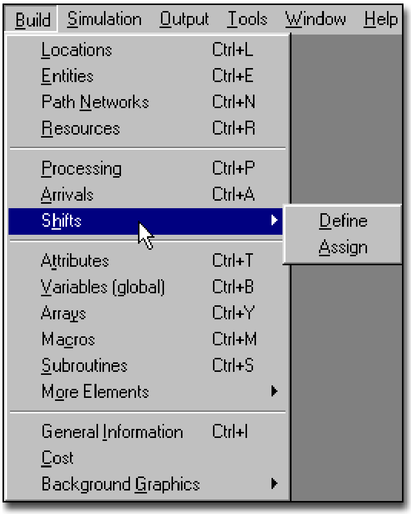
The arrivals logic for the entity called Test1 is as follows:



```
Sample_Type = Test1()
```

Shifts & Breaks

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 have been defined, it may be saved in a shift file with an SFT extension.

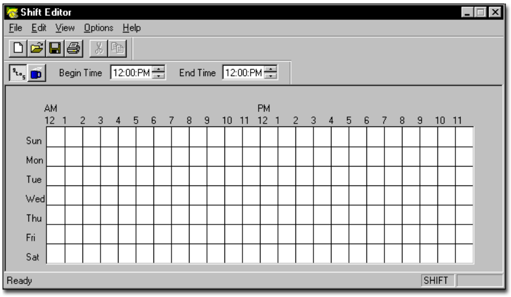


How to define a shift:

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 only be drawn 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.

Please note

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. Dragging 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.

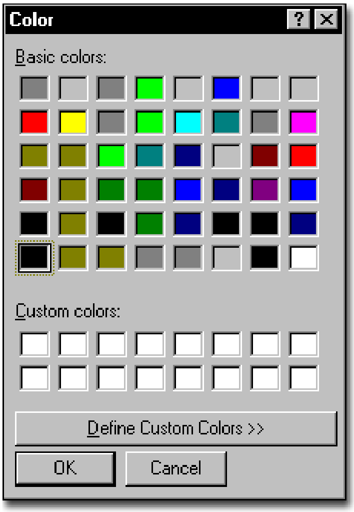
3. In the shift schedule, click the mouse on the day you wish to copy the shift block to.

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**.

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, 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.

You may have an office 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. Simply set up three shifts and assign them all to the office in one step, indicating the appropriate start times for each.

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.

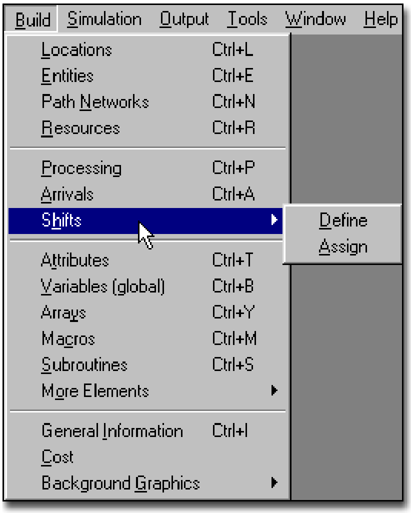
Assigning Shifts

ProModel allows you to can select multiple locations and resources and assign them to a shift in one record. Plus, you can:

- 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.

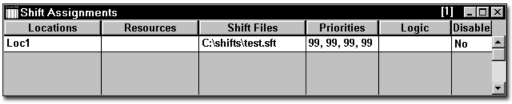
The off-shift and break logic are optional and allow you to control more precisely when a resource or location may go off shift, on break, and how long before it becomes available again.



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:

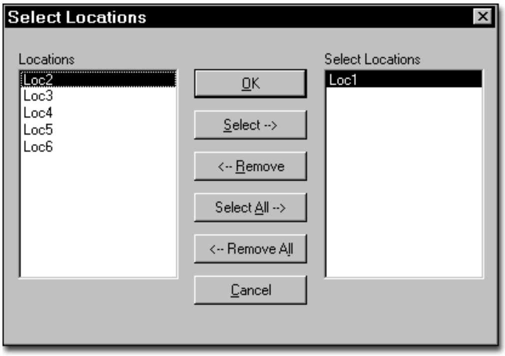
1. Select **Shifts** from the **Build** menu, then click on **Assign**. ProModel displays the Shift Assignments edit window shown below.



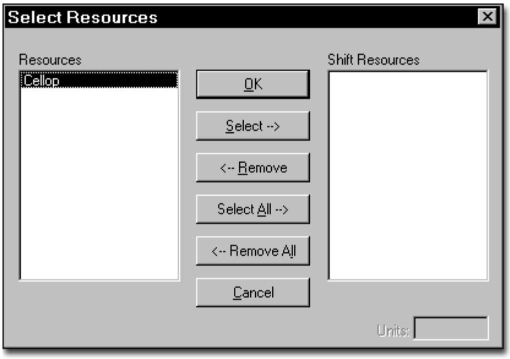
Shift Assignments

Locations	Resources	Shift Files	Priorities	Logic	Disable
Loc1		C:\shifts\test.sft	99, 99, 99, 99		No

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 next*). 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

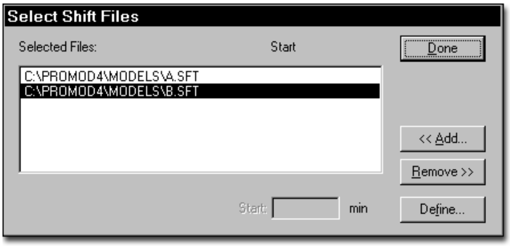
All All units of the resource

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.



Select Shift Files

Selected Files:

Start

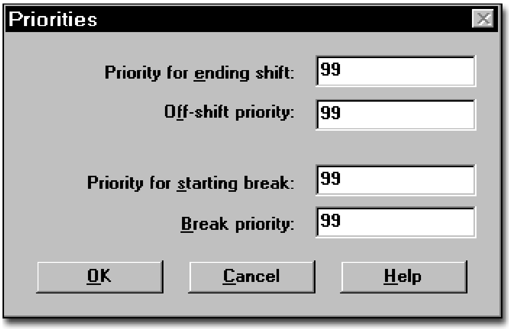
C:\PROMOD4\MODELS\A.SFT
C:\PROMOD4\MODELS\B.SFT

Start: min

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 with 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.



Priorities

Priority for ending shift: 99

Off-shift priority: 99

Priority for starting break: 99

Break priority: 99

These priorities follow standard ProModel priority level and preemption rules. (See *"Locations" on page 96, "Entities" on page 118, and* "Resources" on page 132).

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.

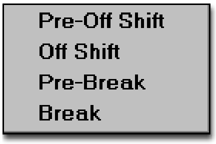
Priority for Starting Break This is the priority for going on break. An entity or downtime must have a higher priority level to prevent this location or resource from going on break at the preset time.

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 to control 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.

How to determine the 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.

Please 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 several functions and statements specifically for shift and break logic: SKIP, PRIORITY, DTLEFT(), FORLOCATION(), and FORRESOURCE(). Following is a brief description of each. For more details, see “Statements and Functions” on page 439.

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 **Applications** is equal to zero before allowing a particular resource called **Loan_Officer** to go off shift. You would enter the following pre-off shift logic:

Pre-off shift logic

```
IF FORRESOURCE() THEN
    BEGIN
        IF RESOURCE() = Loan_Officer THEN
        BEGIN
        WAIT UNTIL Applications = 0
    END
END
```

In addition to these 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.

Shift Downtime Principles

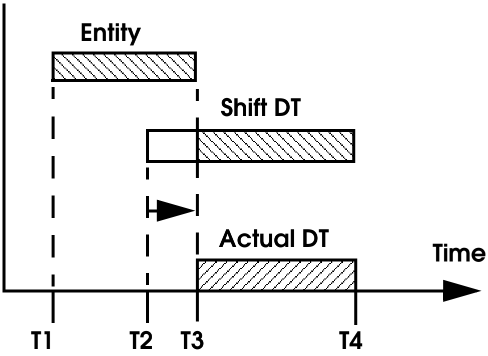
Locations Shift Downtime Principles

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 96 and “Resources” on page 132.

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.

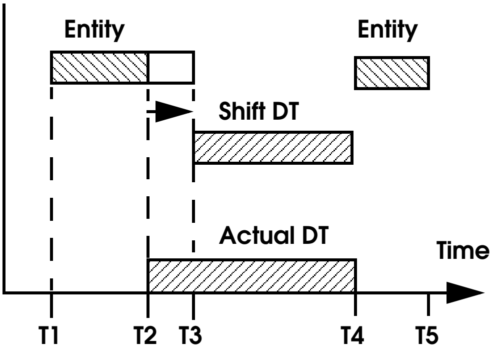
Example 1 (a)



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)



Please 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.

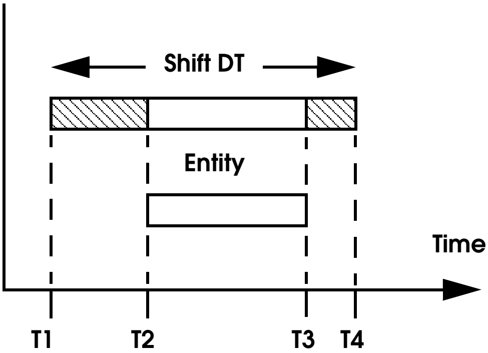
Preempting Off-Shift Locations

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.



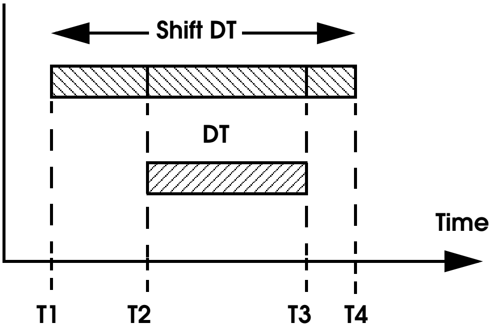
In order for an entity to preempt *any* location downtime (shift or otherwise), it must have a priority level that is at least 2 levels higher than the downtime's priority. In this example the location shift downtime has a priority level of 99 so the entity must have a priority level of 200 or greater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

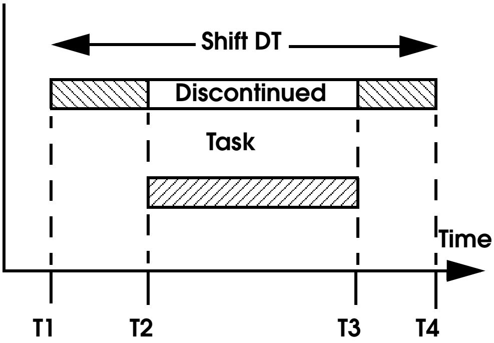
Resource Downtime Principles

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

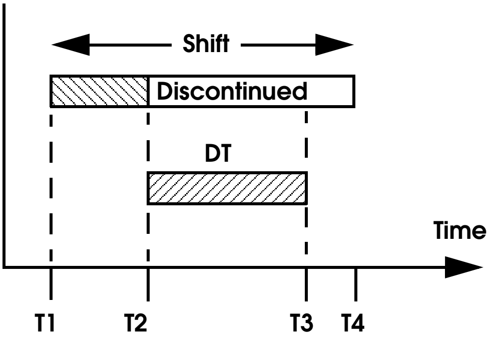
Suppose a resource, repairman, is off-shift. An important machine goes down unexpectedly. Because this machine is a bottleneck in the operation, it is vital to repair the machine as quickly as possible. The repairman is called in and takes 30 minutes to fix the machine. The logic for the downtime to 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

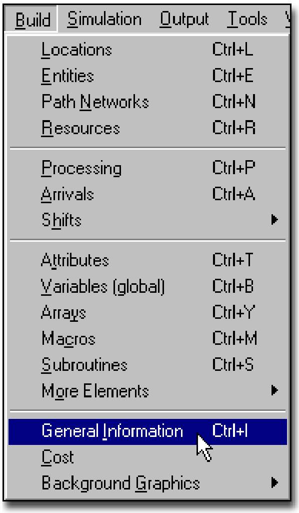
This example shows the effect of a preemptive downtime occurring for a resource already off-shift due to a shift downtime. Since resource downtimes are *not* overlapping, as in the case of location downtimes, the shift downtime in progress is discontinued and the preemptive downtime takes control of 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.

General Information

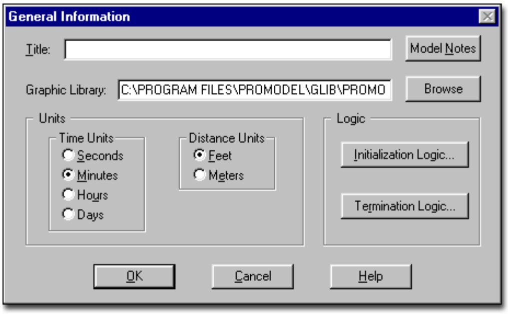
The General Information dialog box allows you to specify basic information about a model, such as its name, default time units, default distance units, and graphic library. You also may specify the model's initialization and termination logic. Finally, a notes window is available for specifying particulars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

General Information Dialog Box


General Information

Title:

Graphic Library: C:\PROGRAM FILES\PROMODEL\GLIB\PROMO

Units

Time Units: ○ Seconds ● Minutes ○ Hours ○ Days

Distance Units: ● Feet ○ Meters

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 467) 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.

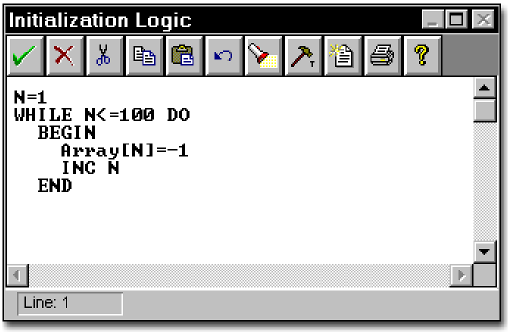
Initialization Logic Opens the Initialization Logic window for specifying initialization logic.

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 180.)

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 180.)

Initialization Logic

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



```
N=1
WHILE N<=100 DO
   BEGIN
      Array[N]=-1
      INC N
   END
```

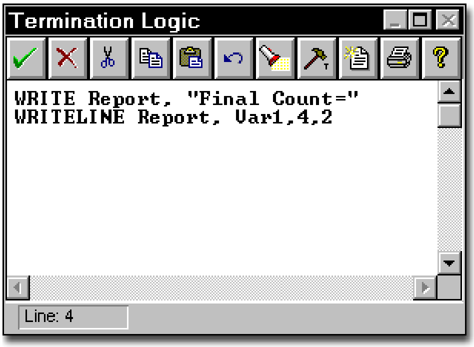
Other common uses of initialization logic include:

- 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 246; also see “Activate” on page 441).

For a discussion of each of the buttons in the Initialization Logic window, see “Editing Logic Windows” on page 78.

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:



```
WRITE Report, "Final Count="
WRITELINE Report, Var1,4,2
```

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.

Please 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. ProModel checks all logic automatically upon selecting OK and, if an error in the logic is found, an error message describing the problem will appear.

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 the “Appendix A” on page 587.

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.

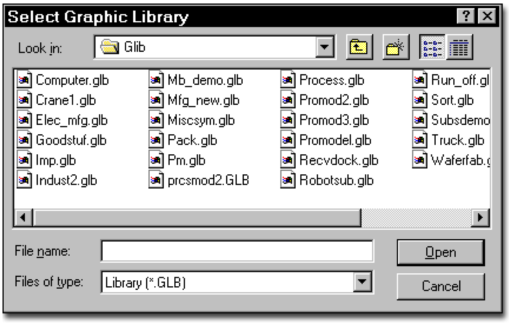
Variables which do not figure into a location's capacity may be initialized in the Initialization Logic without any problem. A variable initialized in the initialization logic could be used as the “First Time” for an arrival or downtime occurrence. This is true because arrivals and downtime occurrences 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 model's graphic library, see “Copying a Graphic from One Library to Another” on page 316.

How to select the desired graphics library:

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



Look in: Glib

File name:

Files of type: Library (*.GLB)

2. Enter the name of the desired graphics library.

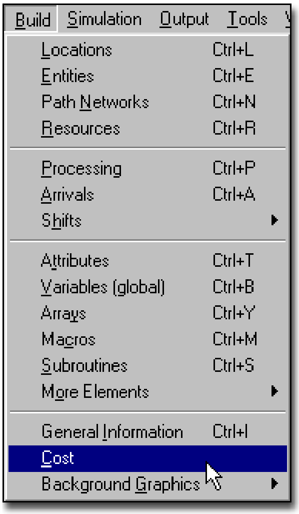
3. Select **OK**.

Please 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 312.

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 includes Costing statistics, automatically generated at the start of the simulation.



How to use cost

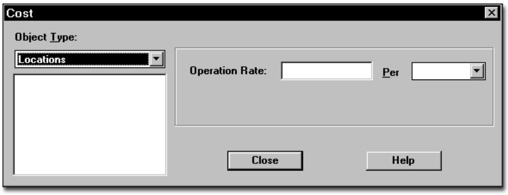
- Choose **Cost** from the **Build** menu. The Cost Dialog appears.

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.

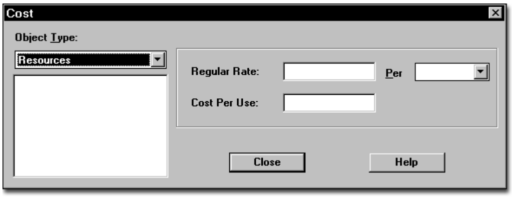
Locations



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 on the SetRate operation statement, see "SetRate" on page 556.

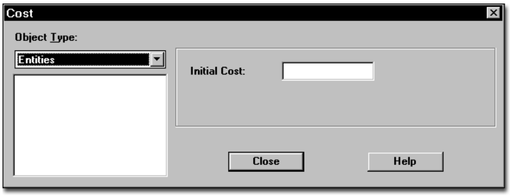
Per This pull-down menu allows you to set the time units for the Regular Rate. Times may be in seconds, minutes, hours, or days as shown here.

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).

Please 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



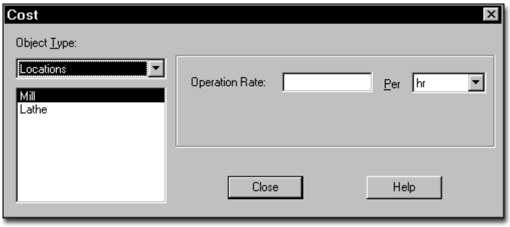
Initial Cost This field allows you to define the initial entity cost for an entity which enters the system through a scheduled arrival, a CREATE statement, or an ORDER statement. ProModel accepts expressions in this field and evaluates them at translation.

Please 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 "IncEnt-Cost" on page 499 for more information.

Building a Model with Costing

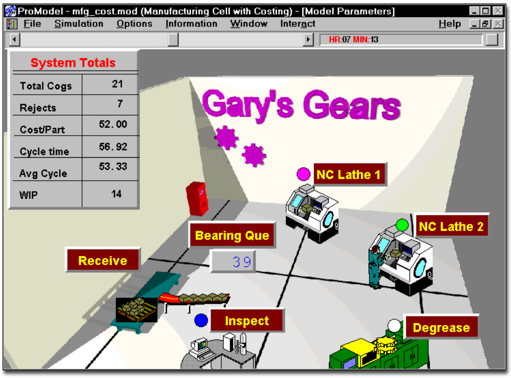
When you build a model using the costing feature, you must first define the locations, resources, and entities used in the model. Once you define these model components, you may assign costing information to them through the Cost option in the Build menu. To collect costing information about your model, 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 187.



Object Type: Locations

Mill
Lathe

Operation Rate: Per hr



HR:07 MIN:13

System Totals	
Total Cogs	21
Rejects	7
Cost/Part	52.00
Cycle time	56.92
Avg Cycle	53.33
WIP	14

Please note

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

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.

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.

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.

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.

Costing Output Statistics

ProModel collects costing statistics only if you uncheck the Disable Cost Statistics option in the Simulation Options menu (see "Enable or Disable Costing" on page 187). Included in the General Statistics Report, ProModel calculates costing statistics.

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)

Please 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

- **Explicit Exits** The number of entities that have explicitly exited. Whenever an entity exits the system, it is an explicit exit except in the following cases:
  - When an entity JOINS or COMBINES with another entity, it implicitly exits the system, and is reported as an exit in the Entity Acitvity report. However, for costing purposes, the entity did not explicitly exit, but its costing information was added to the entity it was JOINED or COMBINED with.
  - When an entity LOADS or GROUPS with another entity, and the entire LOADED or GROUPED entity exits the system, the original entity implicitly exits the system, and is reported as an exit in the Entity Acitvity report. However, for costing purposes, the original entity did not explicitly exit, but its costing information was added to the entire load or group.
- **Total Cost Dollars** 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** % 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.

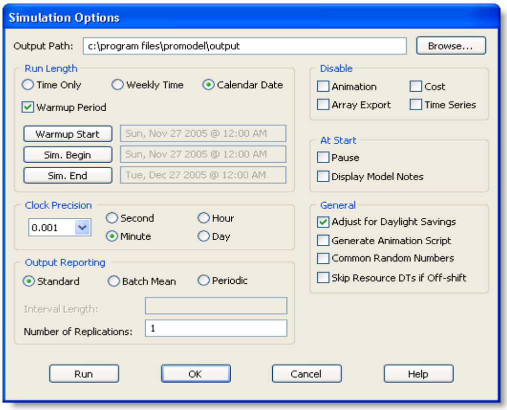
In the above calculations, the rate defined (per day, hour, minute, and second) converts to the default time units specified in the General Information dialog.

Please 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 488 for more information.

Enable or Disable Costing

To *enable* the costing feature, be sure that the Disable Cost option in the Simulation Options dialog is not checked.


Simulation Options

Output Path: c:\program files\promodel\output

Run Length
○ Time Only ○ Weekly Time ● Calendar Date
☑ Warmup Period
Warmup Start: Sun, Nov 27 2005 @ 12:00 AM
Sim. Begin: Sun, Nov 27 2005 @ 12:00 AM
Sim. End: Tue, Dec 27 2005 @ 12:00 AM

Clock Precision
0.001
○ Second ○ Hour
● Minute ○ Day

Output Reporting
● Standard ○ Batch Mean ○ Periodic
Interval Length:
Number of Replications: 1

Disable
□ Animation □ Cost
□ Array Export □ Time Series

At Start
□ Pause
□ Display Model Notes

General
☑ Adjust for Daylight Savings
□ Generate Animation Script
□ Common Random Numbers
□ Skip Resource DTs if Off-shift

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.

Please 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.

Please 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.

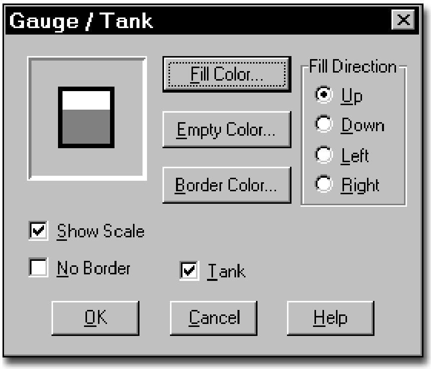
4. Define and reference any necessary tank control subroutines.

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**.



Fill Direction
● Up
○ Down
○ Left
○ Right

☑ Show Scale

□ No Border ☑ Tank

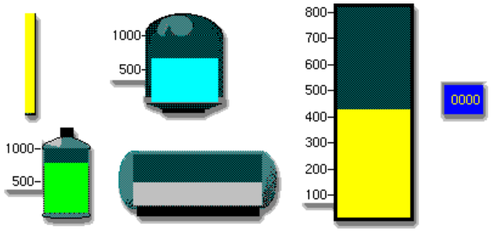
How to change between a tank and a gauge

- 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.



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 pre-defined 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.

Please 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) per time unit defined in the General Information dialog. 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). 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.

Tank_ScheduledDown Similar to the Tank_Down state, except statistics are not collected.

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 to 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 201) 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_GoDownSched subroutine in the off-shift logic using the DTLeft() function as the time parameter. A SKIP statement should follow this function as shown next.

SKIP example

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

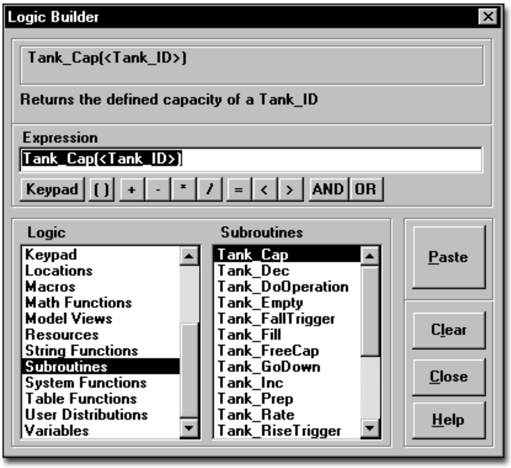
ProModel temporarily suspends tank flow while a tank is down or off shift.

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.

Please 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.



Tank_Cap(<Tank_ID>)

Returns the defined capacity of a Tank_ID

Expression

Tank_Cap(<Tank_ID>)

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 293.

Pre-defined Tank Subroutines

Tank_Fill

Syntax samples

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) to fill into the tank. To fill the tank to capacity, enter Tank_Cap(<Tank ID>).

<Fill Rate>

The rate in units (gallons, pounds) per time unit defined in the General Information dialog. 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 210 and "Initializing and Replenishing Supply Tanks" on page 210.

Tank_Empty

Syntax samples

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) to empty. To empty a tank completely of its current contents, enter Tank_Level [<Tank ID>].

<Empty Rate>

The rate in units (gallons, pounds) per time unit defined in the General Information dialog. 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 //logic repeats continuously
{
WAIT UNTIL
Tank_State(ChemTank)=Tank_Blocked
Tank_Fill(ChemTank, Tank_Level(ChemTank),
60, 0)
}
```

See Also

"Emptying to an Entity" on page 211.

Tank_Transfer

Syntax samples

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) to transfer. To transfer the entire contents of a tank, enter Tank_Level [<FROM Tank ID>].

<FROM Rate>

The rate in units (gallons, pounds) per time unit defined 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) per time unit defined in the General Information dialog 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 212.

Tank_TransferUpTo

Syntax samples

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) per time unit defined in the General Information dialog 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) per time unit defined in the General Information dialog 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 //logic repeats continuously
{
WAIT UNTIL Tank_Level(InProcessTank)<=5000
Tank_TransferUpTo(SupplyTank, InProcessTank, 20000, 100, 0)
}
```

Tank_TransferDownTo

Syntax samples

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) per time unit defined in the General Information dialog 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) per time unit defined in the General Information dialog 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 //logic repeats continuously
{
WAIT UNTIL Tank_Level(TankA)>200
Tank_TransferDownTo(TankA, TankB, 200, 50, 0)
}
```

See Also

"Split Transfers" on page 214.

Tank_SetLevel

Syntax samples

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). 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 210.

Tank_Inc

Syntax samples

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).

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 samples

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).

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 //logic repeats continuously
{
WAIT 15 SEC
Tank_Dec(FillTank, 10)
ORDER 1 Container TO FillStation
}
```

See Also

"Emptying to an Entity" on page 211.

Tank_RiseTrigger

Syntax samples

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)
```

Please 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 215.

Tank_FallTrigger

Syntax sample

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 //logic repeats continuously
{
Tank_FallTrigger(TankX, 1000)
(Insert logic to mix new batch here)
}
```

Please 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 215.

Tank_Cap

Syntax samples

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 samples

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 samples

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 time units defined in the General Information dialog) 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 211.

Tank_GoDown

Syntax samples

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>

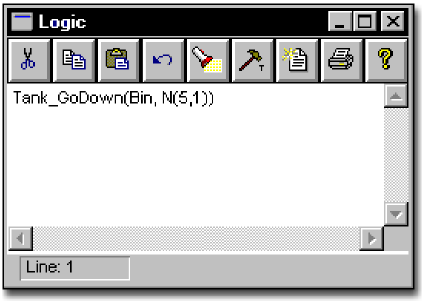
The tank name or location index number.

<Down time>

The duration (in time units defined in the General Information dialog) 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_GoDown(Bin, N(5,1))
```

Tank_GoDownSched

Syntax samples

TANK_GODOWNSCHED (*<Tank ID>, <Down time>*)

TANK_GODOWNSCHED (TankA, 5)

Description

Sets the state of the tank to Tank_ScheduledDown, 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>

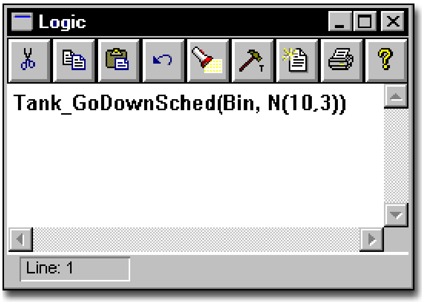
The tank name or location index number.

<Down time>

The duration (in time units defined in the General Information dialog) 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_GoDownSched(Bin, N(10,3))
```

Tank_DoPrep

Syntax samples

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 time units defined in the General Information dialog) 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 //logic repeats continuously
{
(Enter mixing and transfer logic here)
Tank_DoPrep(MixingTank, 30)
}
```

See Also

“Mixing and Reactor Tanks” on page 211.

Tank_SetState

Syntax samples

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 190.

Tank_SelectOutput

Syntax samples

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.)

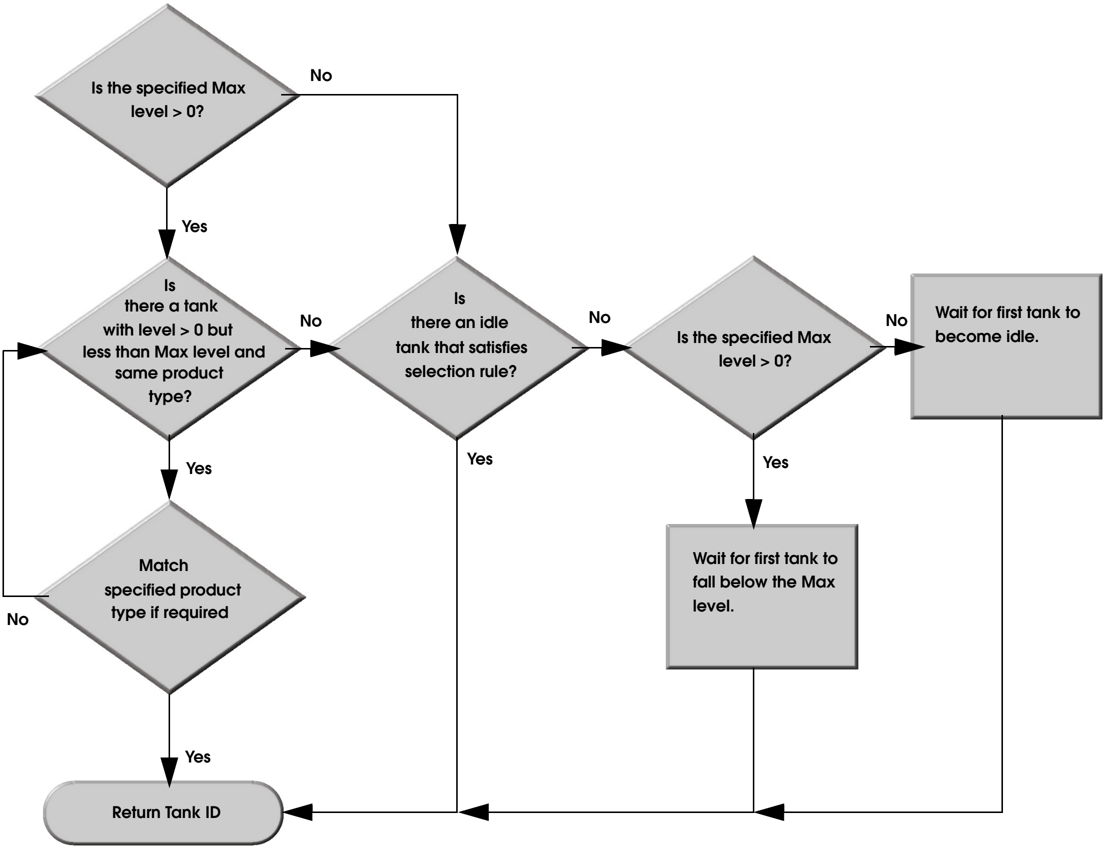
Example

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)
```

The diagram below shows the logic used to make a tank selection:


Is the specified Max level > 0?
No
Yes
Is there a tank with level > 0 but less than Max level and same product type?
No
Is there an idle tank that satisfies selection rule?
No
Is the specified Max level > 0?
No
Wait for first tank to become idle.
Yes
Yes
Yes
Match specified product type if required
No
Wait for first tank to fall below the Max level.
Yes
Return Tank ID

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 213.

Tank_SelectInput

Syntax samples

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 213.

Tank_UpdateStats

Syntax samples

TANK_UPDATESTATS (*<Tank ID>*)

TANK_UPDATESTATS (TankA)

Description

ProModel calls this subroutine automatically whenever you call any of the predefined 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 sample

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 214.

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 time units defined in the General Information dialog. 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 time units defined in the General Information dialog
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 time units defined in the General Information dialog
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 211 for additional information.

Statistics

Please 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 Scheduled 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.

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.

Examples of Tank Control Logic

The following examples show how to model different tank and flow situations. For full models illustrating these situations, see the reference model in the MODELS\REFS directory within the ProModel directory.

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:

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:

Initialize tank level

```
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:

Start with full tank

```
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.

Resume fill with trigger level

```
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:

Mix and clean the tank

```
Tank_Loop //logic repeats continuously
BEGIN
Tank_Transfer (Tank1,MixingTank,2000, 50, 0, 0)
Tank_Transfer (Tank2,MixingTank, 300, 20, 4, 0)
Tank_DoOperation (MixingTank,15) //Mix 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:

Simultaneously mix, then clean tank

```
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.

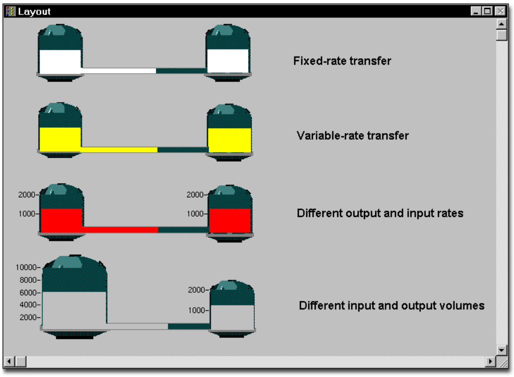
If using the Tank_Dec subroutine, the entity should wait for the fill time *before* decreasing the tank level since Tank_Dec happens *instantly*. For example, if a bottling operation fills a 2 gallon container every 6 seconds, define the following processing logic for the container at the fill station:

Wait to fill from tank, then continue

```
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:

Transfer contents when idle

```
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:

Transfer contents based on condition

```
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 transfer in a pull system

```
Tank_Loop //logic repeats continuously
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
```

Please 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 pre-defined 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.

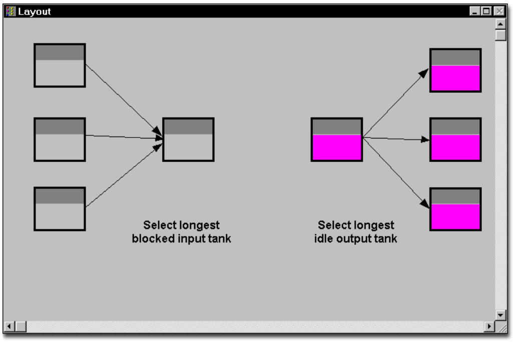
For example, if TankX selects from among three input tanks (Tank1, Tank2, and Tank3) based on which input tank has waited the longest to discharge its contents, you would enter the following logic in the control subroutine defined for TankX:

Select from multiple tanks

```
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*.


Layout
Select longest blocked input tank
Select longest idle output tank

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:

Initiate split transfer

```
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:

Split transfers subroutines

```
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:

Vary the transfer rate

```
IF (Tank_FromID = TankA) AND (Tank_ToID = TankB)
THEN IF Tank_Level(TankB) >= 4000
THEN RETURN 50
ELSE RETURN 150
```

Dynamically Suspending Flow

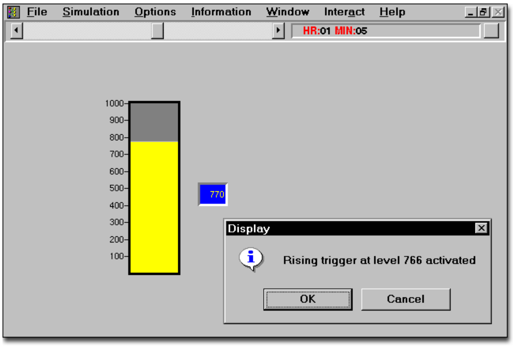
To momentarily interrupt flow into or out of a tank, use the Tank_GoDown subroutine or set the state of the tank to down (Tank_SetState = Tank_Down). This typically happens if a pump fails but may occur in other situations.

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 previous discussion, *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.



Rising trigger at level 766 activated

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:

Trigger subroutine

```
Tank_Loop //logic repeats continuously
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:

Tracking 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:

Define pipes

```
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:

High-rate processing

```
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.

High-rate processing

```
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:

Create new, combined unit

```
Tank Loop //causes logic to repeat continuously

{

Tank_Dec(FillerOutput, 50)

Order 1 Box to Shipping

}
```

Please note

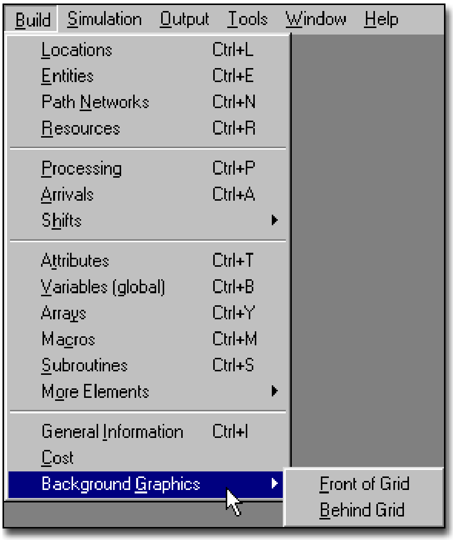
The Tank_Dec statement automatically removes 50 bottles from FillerOutput whenever there are at least 50 bottles available.

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.

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:

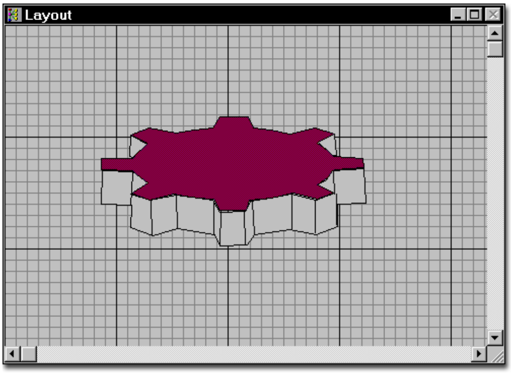
1. Select **Background Graphics** from the **Build** menu.
2. Select **Front of Grid** or **Behind Grid** depending on the mode desired.

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.

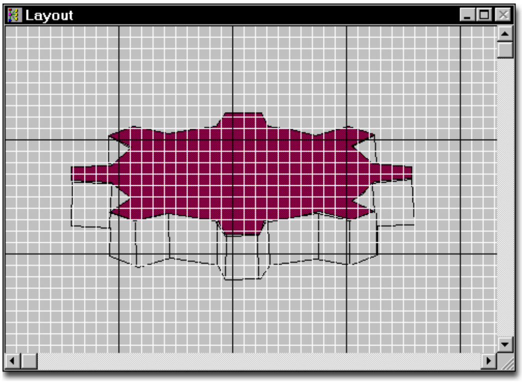
Front of Grid Mode

Creating graphics in this mode places the background graphic in front of the grid as shown below.



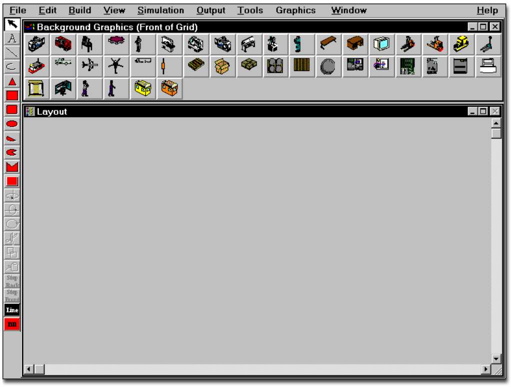
Behind Grid Mode

Creating graphics in this mode places the background graphic behind the grid as shown below.



Background Graphics Editor

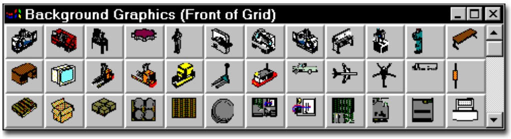
The Background Graphics Editor allows you to place icons, text and other graphic shapes on the layout behind locations and other system element graphics. The arrangement of the two windows and button bar is shown below.



- 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

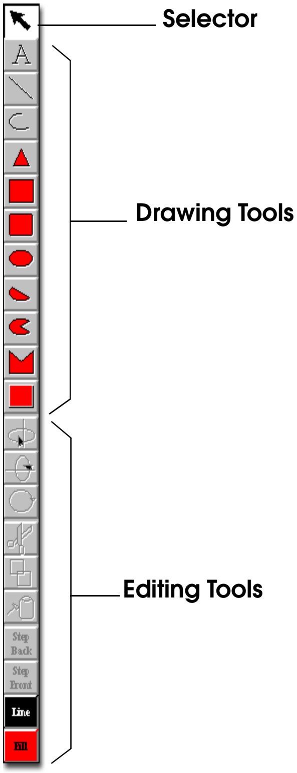
The Library Graphics window contains the icons of the current graphic library file, specified in the General Information dialog. These icons may be placed on the Layout in the same way as other objects.



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.

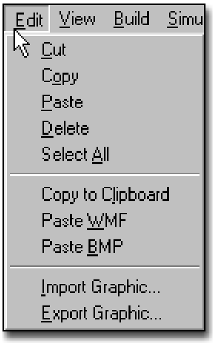


The Background Graphics Editor Tools button bar is the nearly the same Tools button bar used in the Graphic Editor. The only difference is that the Background Graphics Editor Tools button bar does not contain an entity spot or status light tool. For more information on the Tools button bar, see "Graphic Tools Button Bar" on page 323.

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 rectangle 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 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 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.

Please 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 reimported to update the model layout.

Exporting a Graphic

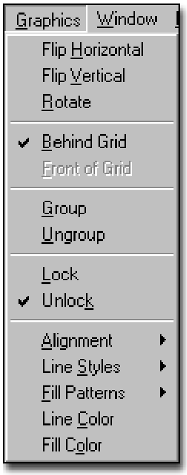
In some cases, it is desirable to export a graphic created in ProModel for use in another application. ProModel will export all objects in the current mode (In Front of Grid and Behind Grid) as one graphic to a WMF or BMP file.

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.

Lock Locks a graphic in place so that it can't be moved. This is helpful for preventing accidental moving of a graphic that you wish to leave stationary.

Unlock Unlocks a locked graphic, allowing it to be moved on the layout.

Alignment When multiple objects are selected on the layout, they can be aligned side to side, top to top, etc.

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.

Please 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 312 for more information on the functions above.

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:

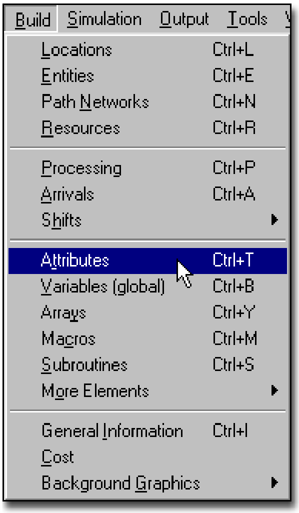
1. Select the graphic on the layout using the selector.

2. Select **Front of Grid** from the **Graphics** menu.

Chapter 6: Building the Model: Advanced Elements

Attributes

Attributes are place holders similar to variables, but are attached to specific locations and entities and usually contain information about that location or entity. Attributes may contain integers or real numbers. You may also assign model element names (e.g., StationA) to an attribute, which is stored as the element's index number but may be referenced by name. Attributes are defined through the Build menu as shown below.



How to create and edit attributes:

1. Select **Attributes** from the **Build** Menu.

Attribute Types

Attributes are classified as follows:

Entity Attributes

Entity attributes are place holders assigned to an entity and contain numerical information about that entity. An entity attribute is identified by its name and may be assigned a value or model element name stored as a value. An entity attribute may be examined and acted upon in any of the following places:

- Arrival logic
- Operation logic
- Move logic which refers to the attribute of the entity being routed
- Min or Max attribute rules for locations and resources
- Routing quantity
- Routing destination priority
- Resource pick up and drop off times
- Entity speed
- Debug condition

Location Attributes

Location attributes are place holders assigned directly to a location and contain numerical information about that location. A location attribute is identified by its name and may be assigned a value or model element name stored as a value. A location attribute may be examined and acted on in any of the following places:

- Arrival logic
- Operation logic
- Move logic
- Min or Max attribute for selecting incoming entities as a location rule
- Routing quantity
- Routing destination priority
- Resource pick up and drop off times
- Location down time logic
- Debug condition

Memory Allocation for Attributes

Because locations always exist during a simulation, location attributes always use memory. However, attributes for entities are not created until the first time any of the entity's attributes are examined or set. At that time, ProModel allocates enough memory for all of the entity's attributes. When the entity exits the system or is absorbed by another entity through a COMBINE or JOIN statement, ProModel automatically frees the memory allocated for its attributes.

Attributes vs. Local Variables

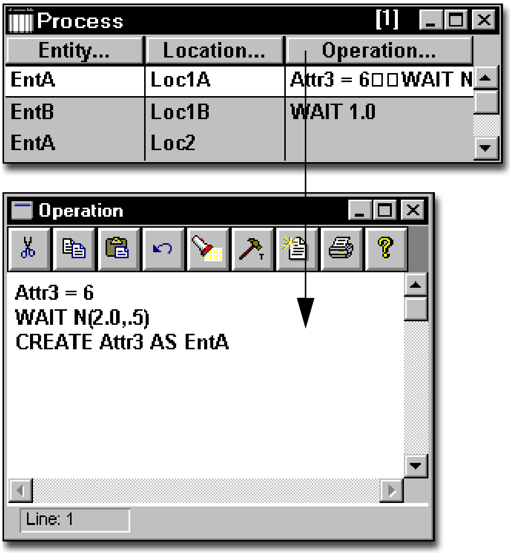
Attributes are primarily useful where the value of the attribute is assigned in one logic section and evaluated in another logic section or field, perhaps at a different location. If, however, an attribute is assigned a value and evaluated within the same logic section (e.g., an operation logic), it would be more appropriate to use a local variable. Local variables act like temporary attributes and are valid only within the logic in which they are defined (see "Variables" on page 231).

Cloning Attributes

Whenever one entity initiates the creation of other entities through a SPLIT AS, CREATE, or ORDER statement, or as the result of specifying multiple outputs in a routing, the attributes of the original entity are automatically copied to each newly created entity.

The following examples show two methods of splitting an incoming batch of material, called BatchA, into one Tote and six EntA's. The first method uses a CREATE statement in the operation logic to create the new entities, called EntA. The original entity, BatchA, will have its name changed to Tote in the routing table. In the second example, both of the new entity types are created in the routing table, so no CREATE statement is needed in the operation logic. In both cases, the attributes attached to the original entity, BatchA, are duplicated in both Tote's and the EntA's attributes.

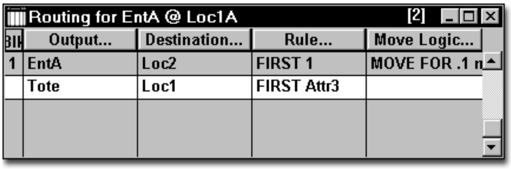
Process Logic



Entity...	Location...	Operation...
EntA	Loc1A	Attr3 = 6□□WAIT N
EntB	Loc1B	WAIT 1.0
EntA	Loc2	

```
Attr3 = 6
WAIT N(2.0,.5)
CREATE Attr3 AS EntA
```

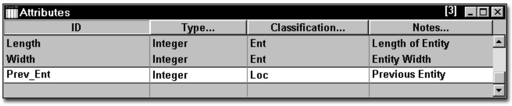
Routing Table



Blk	Output...	Destination...	Rule...	Move Logic...
1	EntA	Loc2	FIRST 1	MOVE FOR .1 n
	Tote	Loc1	FIRST Attr3	

Attribute Edit Table

This edit table is used to define entity and location attributes.



ID	Type...	Classification...	Notes...
Length	Integer	Ent	Length of Entity
Width	Integer	Ent	Entity Width
Prev_Ent	Integer	Loc	Previous Entity

ID The name of the attribute.

Type The type of the attribute, real or integer.

Classification Entity attribute or location attribute.

Notes A general notes field for describing the attribute. Notes fields contain user comments only and are never analyzed by ProModel. Click on the Notes button or double click in the field to open an edit window for entering detailed notes.

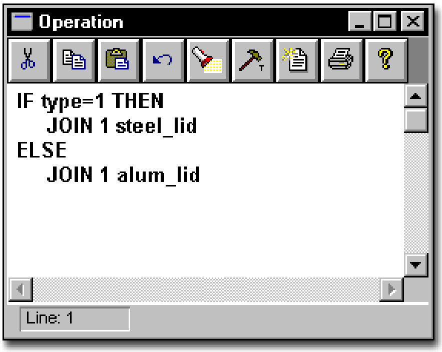
Example of Attributes in Logic

An appliance manufacturer's model contains an assembly location to join lids to pots. The pots are either aluminum or steel and both types of pots arrive at the same assembly location. If an aluminum pot arrives at the assembly location, it must be joined with an aluminum lid. The same is true for a steel pot and lid. The entities, steel_lids and alum_lids, are waiting at a queue to be joined to the pots.

Obviously, one way to model the different pot types is to use two different entity types. This example shows how to achieve the same effect using a single entity type (pot) with an attribute designating whether it is steel or aluminum.

An attribute called "type," defined in the attribute edit table, allows the location to tell what type of pot has arrived at the assembly location. We will use a value of 1 to represent a steel pot and a value of 2 to represent an aluminum pot. When a steel pot enters the system, we assign a value of 1 to the attribute TYPE with the statement TYPE=1. When an aluminum pot enters the system, we set its type to 2.

At the assembly location, we use the following logic:

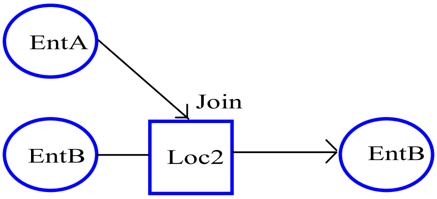


```
IF type=1 THEN
    JOIN 1 steel_lid
ELSE
    JOIN 1 alum_lid
```

This logic checks the type of the pot and then joins a lid according to that type.

Attributes and the JOIN Statement

In some cases, one entity joins to another entity using the JOIN statement (see “Join” on page 504 for more information). If both entities possess attributes before they join together, the resulting joined entity will possess the attribute values of the entity joined to it. In other words, the entity with the JOIN routing rule is effectively destroyed when it gets joined. Consider the following diagram in which EntA joins to EntB:



EntA is joined to EntB.

The logic for the above diagram is as follows:

Process Table

	Location	Operation (min)
EntA	Loc1	Att1 = 1 WAIT 2 min
EntB	Loc2	Att1 = 2 JOIN 1 EntA
EntB	Loc3	...

Routing Table

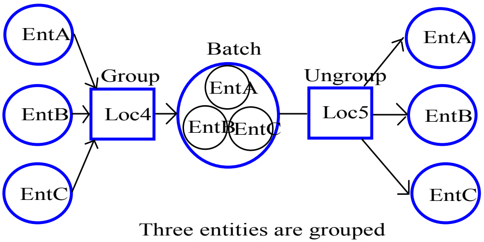
Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	JOIN 1	MOVE FOR 1
1	EntB	Loc3	FIRST 1	MOVE FOR 1
...	...	...	...	...

In the above example, EntB would have an attribute value, Att1, equal to 2 after EntA joined to EntB.

Attributes and the GROUP/ UNGROUP Statements

Suppose several entities, EntA, EntB, and EntC, are grouped together and called Batch (see *“Group” on page 493* and *“Ungroup” on page 568*). Each of the original entities have attributes with values assigned to them before they are grouped. The Batch is processed for 30 minutes, sent to Loc5 and then ungrouped into the original entities.

The attribute values of the individual entities are not transferred to the grouped entity, Batch. In other words, Att1=0 for the entity, Batch. However, once the entities are ungrouped, they retain their original attribute values. The following diagram graphically shows the concept of grouping.



Three entities are grouped together to form a batch which is later ungrouped.

The logic for the diagram is as follows:

Process Table

	Location	Operation (min)
EntA	Loc1	Att1 = 1
EntB	Loc2	Att1 = 2
EntC	Loc3	Att1 = 3
ALL	Loc4	GROUP 3 AS Batch
Batch	Loc4	WAIT 30
Batch	Loc5	UNGROUP
ALL	Loc5	...

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc4	FIRST 1	MOVE FOR 1
1	EntB	Loc4	FIRST 1	MOVE FOR 1
1	EntC	Loc4	FIRST 1	MOVE FOR 1
1	Batch	Loc5	FIRST 1	MOVE FOR 5
...	...	...	...	...

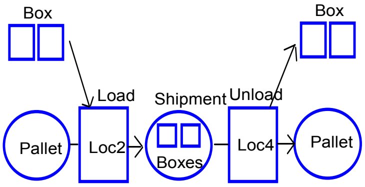
Please note

You can assign an attribute value to a grouped entity. However, once the entities are ungrouped, they retain the attribute values they possessed before they were grouped.

Attributes and the LOAD/ UNLOAD Statements

The LOAD statement loads a specified quantity of entities to the current entity. The loaded entities retain their identity for future unloading through an UNLOAD statement (see “Load” on page 509 and “Unload” on page 571 for more information). When the entities are loaded onto the current entity, the resulting entity retains the attribute value of the current entity.

For example, entities called Box are loaded onto another entity, Pallet. The Boxes are assigned an attribute value, Att1=1. Pallets are also assigned an attribute value, Att1=2. Once the Boxes are loaded onto the Pallet, the loaded pallet is renamed Shipment. The Shipment then has an attribute, Att1=2, because it inherits the attribute value of the Pallet. However, we then assign an attribute value to Shipment, Att1=3. After the Boxes are unloaded from the Pallet, the Boxes retain their original attribute value, Att1=1. Now the Pallet has a different attribute value, Att1=3, which was assigned to the renamed entity, Shipment. Consider the following diagram and logic in which two Boxes are loaded onto a Pallet and renamed Shipment for the output entity:



Two Boxes are loaded onto a Pallet and renamed Shipment in the Output. The Boxes are then unloaded from the Shipment. Shipment is renamed Pallet in the Output.

The logic for the diagram is as follows:

Process Table

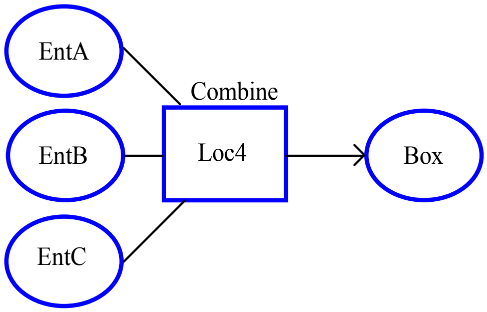
	Location	Operation (min)
Box	Loc1	Att1 = 1 WAIT 2 min
Box	Loc2	Att1 = 2 LOAD 2
Shipment	Loc3	Att1 = 3 WAIT 20
Shipment	Loc4	UNLOAD 2
Box	Loc4	WAIT 10

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Box	Loc2	LOAD 1	MOVE FOR 1
1	Ship-ment	Loc3	FIRST 1	MOVE FOR 3
1	Ship-ment	Loc4	FIRST 1	MOVE FOR .5
1	Pallet	Loc5	FIRST 1	MOVE FOR 1
1	Box	Loc5	FIRST 1	MOVE FOR 2

Attributes and the COMBINE Statement

Consider the example where several entities are combined permanently into a single entity, Box (see *"Combine" on page 459* for more information). The combined entity, Box, assumes the attribute values of the last entity that was combined to the single entity. If three entities, EntA, EntB, and EntC, are combined to form a single entity called Box, and EntC was the last entity that was combined, the Box will have the same attribute values as EntC. Therefore, if EntC had an attribute (Att1=5), then Att1=5 for the combined entity, Box. The following diagram demonstrates three entities combining into one entity.



Three entities are combined to form a single entity called Box.

The logic for the diagram is as follows:

Process Table

	Location	Operation (min)
EntA	Loc1	Att1 = 1 WAIT 2 min
EntB	Loc2	Att1 = 3 WAIT 3 min
EntC	Loc3	Att1 = 5 WAIT 6
ALL	Loc4	COMBINE 3
ALL	Loc5	...

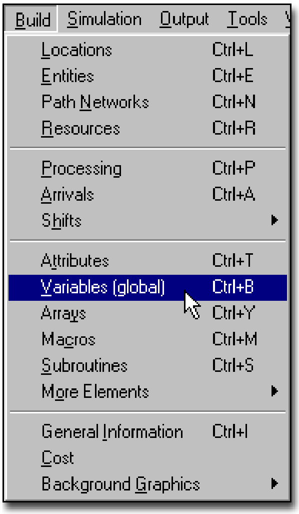
Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc4	FIRST 1	MOVE FOR 1
1	EntB	Loc4	FIRST 1	MOVE FOR 3
1	EntC	Loc4	FIRST 1	MOVE FOR .5
1	Box	Loc5	FIRST 1	MOVE FOR 1
...	...	...	...	...

In the above example, EntC is the last entity to be combined so the entity Box assumes the attribute value of EntC, Att1=5.

Variables

Variables are of two types: global and local. Global variables are place holders defined by the user to represent changing numeric values. Local variables are place holders which are available only within the logic that declared them. Variables can contain either Real numbers or Integers, including element index values, and are typically used for decision making or recording information. A global variable can be referenced anywhere numeric expressions are allowed in a model. If a variable or attribute is needed only in a single block of logic, it is easier to define a local variable right inside the logic block. Global variables are defined in the Variables Editor, accessed from the Build menu. Local variables are defined with the INT and REAL statements. (See “Local Variables” on page 233 for more information.)

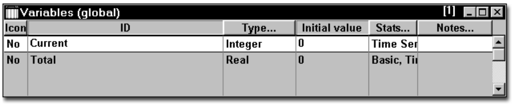


How to access the variable edit table:

1. Select **Variables** (global) from the **Build** menu.

Variable Edit Table

This edit table is used to define Variables used globally in the model. A description of each field is given below.


Variables (global)

Icon	ID	Type...	Initial value	Stats...	Notes...
No	Current	Integer	0	Time Ser	
No	Total	Real	0	Basic, Ti	

Icon This field shows “Yes” if an icon for the variable appears on the layout. A variable’s icon looks like a counter and displays the variable’s value.

ID The variable’s name.

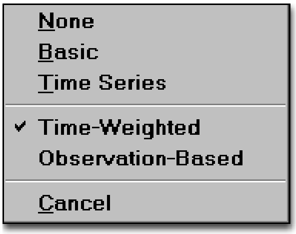
Type The type of variable, real or integer.

Initial Value The initial value of the variable to be assigned at the start of the simulation. By default, initial values are 0, but can be changed in the edit table to whatever value you want. Any expression can be entered here (including previously defined variables) except attributes and system functions.

Stats ProModel collects statistics for each variable on three levels of detail, None, Basic, and Time Series.

- **None** No statistics are collected for this variable during simulation.
- **Basic** Collects basic statistics such as total changes, average minutes per change, current value, and average value.
- **Time Series** Collects all the basic statistics plus the value history based on time or observations. When you select Time Series, ProModel collects either time-weighted or observation-based statistics

for the variable depending upon the type selected.



Variable Observation Record	Value	Time in Hours
Observation 1	6	1
Observation 2	5	2
Observation 3	6	3
Observation 4	5	4
Total	**22**	**10**

Time-Weighted Collects information on the percentage of time the variable was at a specific value. As shown in the above table, the average value of the variable is:

$$\frac{(6 \text{ x } 1) + (6 \text{ x } 3) + (5 \text{ x } 2) + (5 \text{ x } 4)}{10} = 5.4$$

Observation-Based Collects information on the number of times the variable changed to a specific value. As shown in the above table, the value of the variable is 6, then 5, then 6, then 5. The average would simply be:

$$\frac{6 + 5 + 6 + 5}{4} = 5.5$$

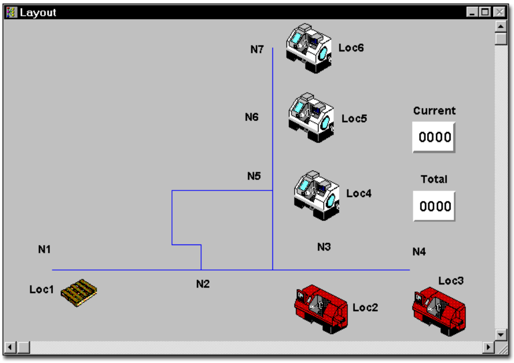
Notes A general notes field for describing the variable. Click on the Notes button or double click in the field to open an edit window for entering detailed notes.

Please note

In order to create plots or histograms for a variable, Time Series stats must be selected in the variables edit table.

Variable Layout

An icon to show a variable's value during a simulation may be placed anywhere on the layout. The window below shows the icons for the variables Current and Total at the right side of the screen. Each icon has been labeled with a background graphic.



How to place an icon for a variable on the layout:

1. Highlight the desired variable in the Variable edit table.

2. Click on the layout where the icon is to appear.

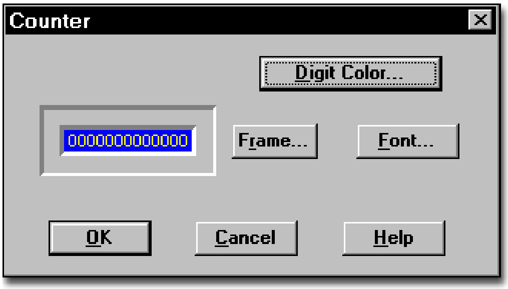
3. Size the icon by dragging an edge or corner of the sizing box.

How to remove an icon for a variable from the layout:

1. Double click on the icon.
2. Choose **Delete** from the resulting menu. The icon is removed from the layout, but the variable remains in the model.

Editing a Variable's Icon

A variable's icon can be customized as necessary by simply double clicking on the icon and choosing Edit. A dialog box appears as shown below for specifying the characteristics of the variable icon or counter.



How to edit a variable's icon:

1. Double click on the icon.
2. Click on the **Digit Color**, **Frame**, or **Font** buttons to adjust the respective setting.
3. Click on **OK**.
4. All other variable icons that you create from now on will retain these modifications.

Local Variables

Local variables function as though they were temporary attributes defined in a specific logic section which disappear when the logic section is finished executing. Local variables are useful for test variables in loops and storing locally used, unique values for each entity at the current location.

Local variables are used within a block of logic (i.e., operation logic, subroutines) and are declared with an INT or REAL statement. Local variables are only available within the logic in which they are declared and are not defined in the Variables edit table. A new local variable is created for each entity to encounter an INT or REAL statement (See “Int” on page 502 and “Real” on page 539). A local variable is specific to each entity, in much the same way an attribute is specific to an entity, except that the local variable is only available while the entity processes the logic to declare the local variable. Local variables may be passed to subroutines as parameters. Local variables are available to macros.

Example

A plant manufactures valves of 10 different sizes, such as 10", 20", ... All valves are inspected at a common inspection station and then move to a dock where they are loaded onto pallets. The pallets are designed to hold only a certain size valve. Therefore, a pallet designed to hold 10" valves can only hold 10" valves, not 20" valves.

Suppose a Pallet enters a multi-capacity location, Dock. Each Pallet has a different entity attribute, p_type, describing the type of valve it can hold. Valves are loaded onto the Pallet. The 10” valves must be loaded onto the pallet designed to hold the 10” valves. Therefore, the attribute value of the Valve, v_type, must match the attribute value of the Pallet, p_type. We can use local variables to accomplish this modeling task. The logic is as follows where X is a local variable:

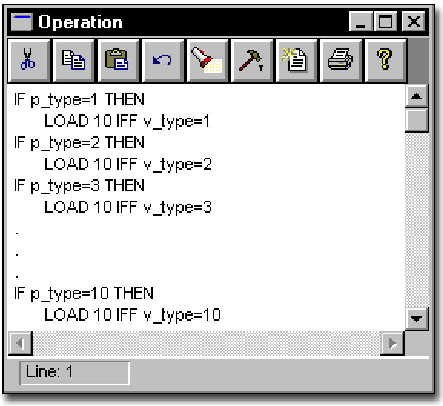
Process Table

	Location	Operation (min)
Valve	Inspect	WAIT 5
Pallet	Dock	INT X X = p_type LOAD 10 IFF X = v_type WAIT 10

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Valve	Dock	LOAD 1	MOVE FOR 2
1	Pallet	Delivery	FIRST 1	MOVE FOR 8

If we had not used local variables, we would need to use the following operation logic for Pallet at Dock:



```
IF p_type=1 THEN
    LOAD 10 IFF v_type=1
IF p_type=2 THEN
    LOAD 10 IFF v_type=2
IF p_type=3 THEN
    LOAD 10 IFF v_type=3
.
.
.
IF p_type=10 THEN
    LOAD 10 IFF v_type=10
```

As can be seen from the two examples of logic, the first example is much easier and more straight forward.

It is important to note that using “LOAD 10 IFF p_type = v_type” in the operation logic would not work for the intended purpose. Attributes referenced in IFF conditions always refer to the entity being loaded. Set the value of a local variable, X, to the pallet attribute, p_type, so it can be referenced in the LOAD statement. The pallet attribute cannot be directly referenced in the LOAD statement.

If Dock was a single capacity location, using a global variable would work the same as using a local variable. However, because Dock is a multi-capacity location, it can load valves onto multiple pallets at the same time. If a global variable was used instead of a local variable, the global variable would change each time a pallet entered Dock. If there were two different types of pallets at Dock, there would be only one type of valve loaded on the pallet because the global variable refers to both pallets.

Suppose, for example, a global variable, type, signifies the pallet attribute, p_type. We assign type=p_type at the beginning of the operation logic for location Dock. The first pallet arrives and type=3. Therefore, only valves with v_type=3 are loaded onto the pallet. Another pallet enters Dock and type=5. Now only valves with valve_type=5 are loaded onto both pallets.

Please note

Local variable notes:

1. You may not use the WAIT UNTIL statement with local variables.

2. The local variable definition only needs to appear somewhere in the logic before being referenced. The entity does not need to execute the local variable definition statement (INT, REAL).

Arrays

An array is a matrix of cells that contain real or integer values. Each cell in an array works much like a variable, and a reference to a cell in an array can be used anywhere a variable can be used. A one-dimensional array may be thought of as a single column of values. A two-dimensional array is like having multiple columns of values (similar to a spreadsheet).

To reference a cell in a one-dimensional array, give the name of the array and enclose the cell number in brackets. For example, if a one-dimensional array containing ten cells is named OpnArray, to reference the fifth cell you would use OpnArray[5]. To reference a cell in a two-dimensional array, simply use the name of the array with the row and column number in brackets. For example, the statement OpnArray[3,4] references a cell on the third row in the fourth column.

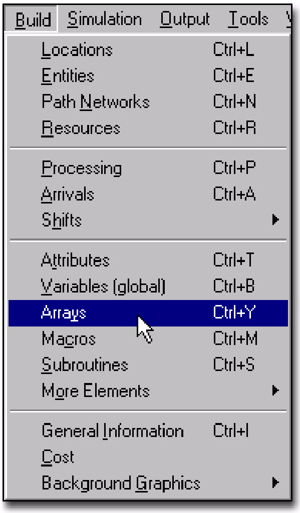
Arrays with more than two dimensions are more difficult to picture, but work exactly the same as one and two dimensional arrays. For example, if an array has a third dimension, you can reference any cell simply by adding a comma before the number of the desired cell in the third dimension. For example, Tool[3,5,8]. Four and five-dimensional arrays work the same way. The maximum number of dimensions for an array is 20.

Array cell values are assigned in the exact same way as you would assign a value to a variable. For example, to assign the value 18 to the cell at the fifth row and second column in an array named Arr1 you would use the following statement.

Assign array cell value

Arr1[5,2]=18

Arrays are defined in the Arrays editor accessed through the Build menu.



How to use the arrays editor:

1. Select **Arrays** from the **Build** menu.

Example Arrays

The following examples show how elements are referenced in a one-dimensional array with five cells and in a two-dimensional array with fifteen cells.

One-dimensional array

OpnArray[5]

Cell[1]
Cell[2]
Cell[3]
Cell[4]
Cell[5]

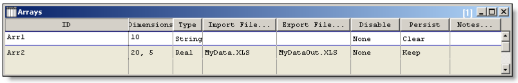
Two-dimensional array

Tool[3,5]

Cell[1,1]	Cell[1,2]	Cell[1,3]	Cell[1,4]	Cell[1,5]
Cell[2,1]	Cell[2,2]	Cell[2,3]	Cell[2,4]	Cell[2,5]
Cell[3,1]	Cell[3,2]	Cell[3,3]	Cell[3,4]	Cell[3,5]

Arrays Edit Table

The Arrays edit table is used to define Arrays that are used in the model. The fields of the Arrays edit table are explained on the following page.



ID	Dimensions	Type	Import File...	Export File...	Disable	Persist	Notes...
Arr1	10	String			None	Clear	
Arr2	20, 5	Real	MyData.XLS	MyDataOut.XLS	None	Keep	

ID: The name of the array.

Dimensions: The size of each dimension of the array in cells. For example, the dimensions of a one-dimensional array of 100 cells is “100.” Likewise, a two-dimensional array with 50 rows and five columns would have dimensions of “50,5.” The number of rows is first, followed by a comma and then the number of columns. An array cannot have more than 20 dimensions.

Type: The type (integer, real, string, or expression) for all cells in the array.

Import File: The name of the spreadsheet from which you will populate the array. You may enter either a fully qualified path to your .xls file or a path relative to your .mod file. For example, if your .xls file is in the same directory as your .mod file, simply enter the name of the .xls file into this field.

See “Import Data into Arrays” on page 237 for more information on using the Import dialog.

Export File: The name of the spreadsheet to which you will save the array data. You may enter either a fully qualified path to your .xls file or a path relative to your .mod file. For example, if your .xls file is in the same directory as your .mod file, simply enter the name of the .xls file into this field.

See “Export Arrays to Spreadsheets” on page 239 for more information on using the Export dialog.

Disable: Use this option to have the import file, export file, or both ignored during simulation without deleting the name of the import or export files from the Array record. Choose “None” to use the specified import and export files.

Persist: When running multiple replications you may choose to keep the values in the array from one replication to the next or clear (reset) the array values every replication.

For example, if you run a simulation for three scenarios with an array that “Keeps” its values, the data in the array at the end of the first replication will be kept and used as the second replication begins. If you are using an Import File, it will be used to populate the array for the first replication, but ignored for subsequent replications when “Keep” is selected for the array.

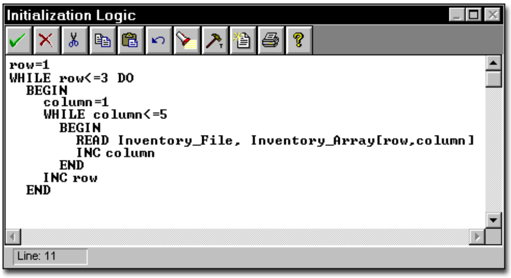
If you choose to “Clear” the array, no array information will be kept from one replication to another and the array will be reset with its initial information at the beginning of each replication (see the next heading “Initializing Arrays”). If you are using an Import File, it will be used to populate the array at the beginning of each replication when “Clear” is selected for the array.

Notes A general notes field for entering descriptive information about the array. Click the heading button or double click in this field to open a larger window for entering notes.

The window pictured above shows how to define the example arrays that appear on the previous page.

Initializing Arrays

By default, all cells in an array are initialized to zero. Initializing cells to some other value should be done in the initialization logic. A WHILE...DO loop is useful for initializing array cell values. The logic below fills a 3 x 5 array (3 rows and 5 columns) with values from an external, general read file.



```
row=1
WHILE row<=3 DO
  BEGIN
    column=1
    WHILE column<=5
      BEGIN
        READ Inventory_File, Inventory_Array[row,column]
        INC column
      END
    INC row
  END
```

This example uses the variables Column and Row (which may be defined as local variables) along with two WHILE...DO loops to assign every cell in the array Inventory_Array a value from a general inventory file. The logic first sets the value of the variable Column to one. It then assigns all the cells in column one a value by reading a value from the external file and incrementing the variable Row. When all the cells in column one have a value, the logic increments to the second column and does the inside loop again. It repeats this loop until each cell in the array has a value.

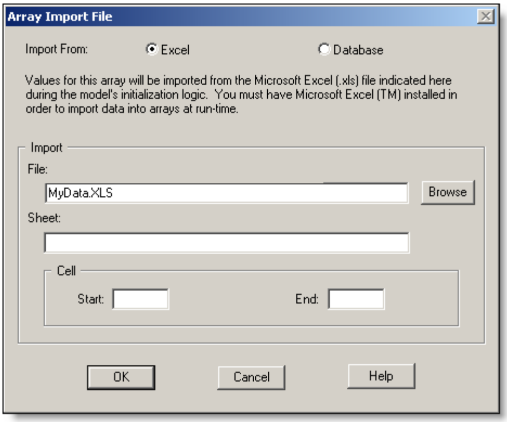
Please note

Assigning values to a cell in an array can be done in any expression or logic field, such as initialization and operation logic. However, arrays cannot be used in logic elements that determine a model's structure, such as location capacity. See "Execution Time of Initialization and Termination Logic" on page 180 for a list of logic elements used to define model parameters.

Import Data into Arrays

When you import data, from either an external Excel spreadsheet or SQL database, into an array, ProModel loads the data from left to right, top to bottom. Although there is no limit to the quantity of values you may use, ProModel supports only *two-dimensional* arrays.

Import from an Excel Spreadsheet



Import From: ● Excel ○ Database

Values for this array will be imported from the Microsoft Excel (.xls) file indicated here during the model's initialization logic. You must have Microsoft Excel (TM) installed in order to import data into arrays at run-time.

Import

File: MyData.XLS

Sheet:

Cell

Start: End:

Import File The name of the spreadsheet you will use. You may enter either a fully qualified path to your .xls file or a path relative to your .mod file. For example, if your .xls file is in the same directory as your .mod file, simply enter the name of the .xls file into this field.

Sheet Name The name of the sheet from which you will import the array data.

Please note

If your spreadsheet contains only a single data set, ProModel *will automatically load the data* *into the array—you do not have to define any cell information unless you wish to limit the contents of the array to a portion of the data set.*

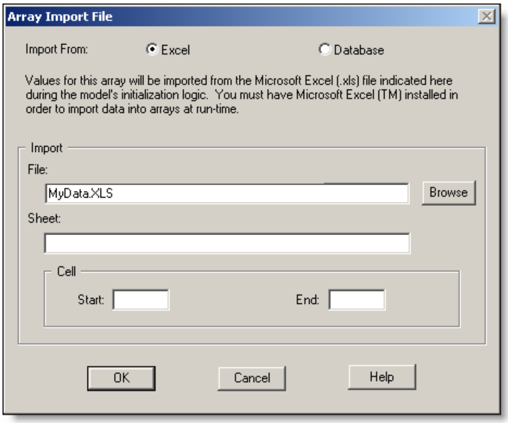
Array import requires Microsoft Excel 97 or later.

Import Start Cell The *first* piece of data to place into the array.

Import End Cell The *last* piece of data to place into the array.

How to import Excel spreadsheet data into array:

1. Select **Arrays** from the **Build** menu.
2. Select or create the array record (row) you wish to import.
3. Click the **Import File** button on the Arrays dialog.
4. In the **File** field, enter the name of the spreadsheet you wish to use (you may also browse to select a file).



Array Import File

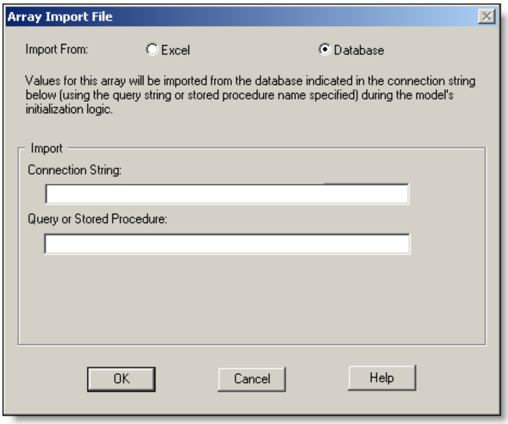
Import From: ● Excel ○ Database

Values for this array will be imported from the Microsoft Excel (.xls) file indicated here during the model's initialization logic. You must have Microsoft Excel (TM) installed in order to import data into arrays at run-time.

File: MyData.XLS

5. In the **Sheet Name** field, enter the name of the worksheet that contains the data you wish to use.
6. Enter the **Import Start Cell** location. The value in this cell will occupy the *first* position in the array.
7. Enter the **Import End Cell** location. The value in this cell will occupy the *last* position in the array.
8. Click **OK**.

Import from a SQL Database



Array Import File

Import From: ○ Excel ● Database

Values for this array will be imported from the database indicated in the connection string below (using the query string or stored procedure name specified) during the model's initialization logic.

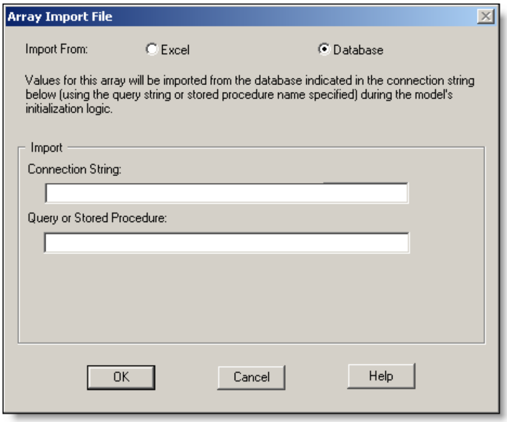
Connection String The connection string used to connect to the SQL database including all necessary security parameters.

Query or Stored Procedure Enter the SQL query or the name of the stored procedure to be executed during the model's initialization logic to populate the array.

How to import SQL database data into array:

1. Select **Arrays** from the **Build** menu.
2. Select or create the array record (row) you wish to import.
3. Click the **Import File** button on the Arrays dialog.

4. Click on the Database radio button.

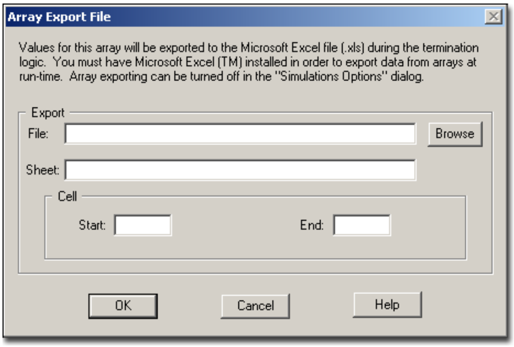

Values for this array will be imported from the database indicated in the connection string below (using the query string or stored procedure name specified) during the model's initialization logic.

5. In the **Connection String** field enter the necessary string to connect to the desired database.

6. In the **Query or Stored Procedure** field enter the SQL query or stored Procedure name that will be used to populate the array during the initialization logic.

7. Click **OK**.

Export Arrays to Spreadsheets


Values for this array will be exported to the Microsoft Excel file (.xls) during the termination logic. You must have Microsoft Excel (TM) installed in order to export data from arrays at run-time. Array exporting can be turned off in the "Simulations Options" dialog.

The fields in the Export Arrays dialog have similar functionality as those for the Import Arrays dialog.

If you export multiple times to the same spreadsheet, ProModel will overwrite the spreadsheet with new data. If you wish to prevent your data from being overwritten, you can make a backup of the spreadsheet between each running of the simulation.

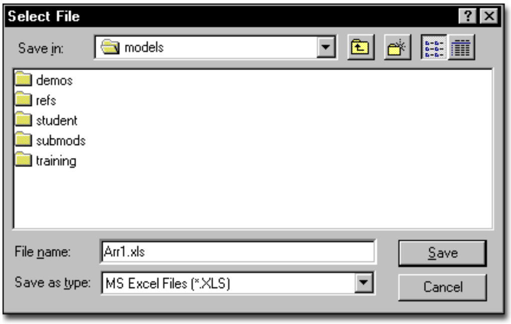
If you are running multiple scenarios and/or replications, see step 5 below for a description of how the spreadsheet handles the multiple scenarios/replications.

How to export array data:

1. Select **Arrays** from the **Build** menu.

2. Click the **Export File** button on the Arrays dialog.

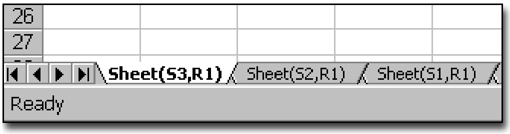
3. Enter the name of the spreadsheet file you wish to use, or browse to select a file.


File name: Arr1.xls

Save as type: MS Excel Files (*.XLS)

4. Click **OK**.

5. When the simulation is run, the array data will be saved to the spreadsheet file. If you ran multiple scenarios and/or replications, data from each one will be saved to a separate sheet in the spreadsheet file, as seen below.



Using Arrays

Using arrays can simplify a model. Suppose you need to model an assembly line that attaches components to a computer motherboard. Furthermore, you want to track the usage of component parts over time. Without an array, hundreds of individual entities of various types would have to represent hundreds of individual components. Keeping track of all the components would be very complex, not to mention all of the join operations and routings for performing the assembly. Instead, various cells in a one-dimensional array could track the number of each type of component used during the simulation.

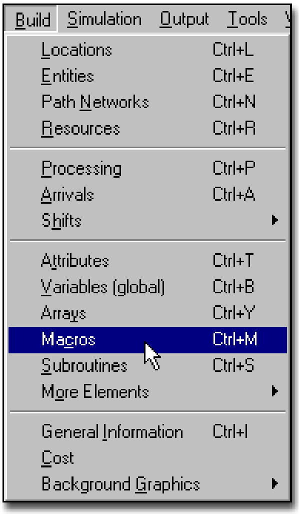
An array can do the job more efficiently. The initial inventory level for each component could be stored in an external file and read into the cells of the array at the start of the simulation. The first cell might contain the inventory level of transistors; the second could contain the inventory level of capacitors and so on. When a motherboard arrives at the location adding the components, each cell's value is decremented according to the number of that type of component joined to the motherboard. If each motherboard requires twelve transistors and five capacitors, then every time a motherboard arrives at the location, the array's first cell is reduced by twelve and the second cell is reduced by five. Thus the model becomes much less complex because it requires fewer entities and less logic.

Notes on Arrays

1. If a warm-up time is specified, array values are not reset.
2. Arrays can be nested. For example, if Arr1[2,3] is equal to three, then the statement Arr2[5,Arr1[2,3]] works exactly like the statement Arr2[5,3].
3. You can examine the value of a cell in an array during a simulation by choosing Information and then Arrays. This information can also be printed.
4. Arrays can be used with the WAIT UNTIL statement.
5. Statistics are not generated for arrays. However, if you would like to see the final value of an array's cell, you can use the array export feature to export to Excel, you can place a PAUSE statement in the termination logic and then view the array under the Information menu, or you could print an array's values or write them to an external file as part of the termination logic. If you want more statistical information on a particular cell, assign the cell to a variable and then choose basic or time-series statistics for the variable.

Macros

A macro is a place holder for an often used expression, set of statements and functions, or any text that might be used in an expression or logic field. A macro can be typed once, and then the macro's name can be substituted for the text it represents anywhere in the model and as many times as necessary. Macros are defined in the Macros Editor, accessed from the Build menu.

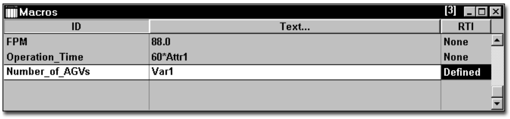


How to create and edit macros:

1. Select Macros from the Build Menu.

Macro Editor

The Macro edit table is used to assign recurring text to a reference name.



ID	Text...	RTI
FPM	88.0	None
Operation_Time	60*Attr1	None
Number_of_AGVs	Var1	Defined

ID A name to identify the macro.

Text Any text to be substituted where the macro name is called. This text may be a complete expression, an entire logic block, or even part of a logic block.

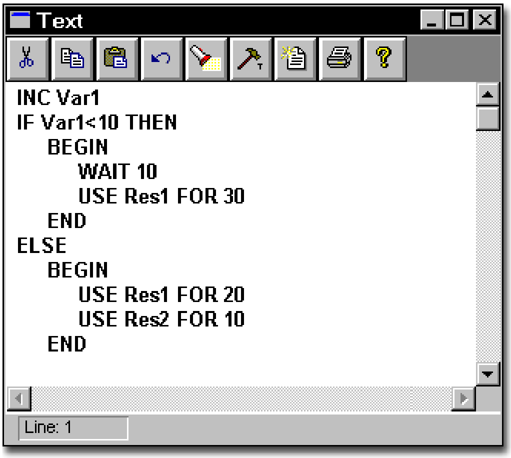
Options Allows you to define the macro as a run-time interface parameter or select a resource group.

The example table above defines three macros. The first macro is simply a numeric constant, with fpm representing a conversion factor from miles per hour to feet per minute. If a number is used in multiple places in a model, then a macro makes it possible to change that number throughout the model simply by changing the macro itself. The second macro, Operation_Time, calculates the various operation times at different locations depending on the attributes at the locations. The last macro, Number_of_AGVs, is a run-time interface variable used to define the number of AGVs in the simulation model (see "Run-Time Interface" on page 242).

A macro is different from a subroutine because a macro cannot pass or return a value. However, because it is simply a text replacement, a macro can reference any expression valid in the expression or logic field that called the macro. For instance, the string "the number of entries is" might be a macro called mac1. This macro by itself is not a valid expression. However, when used with the DISPLAY statement in the operation logic (i.e., DISPLAY mac1), the compiler will recognize the macro as a string.

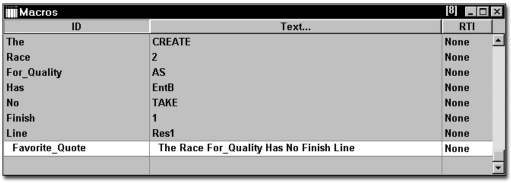
A macro may be used in any expression field, but may only contain a numeric expression (e.g., Entries (LOC1), U(5,1), Var1+Att2, etc.). In addition, a macro used in an expression field may not contain multiple lines of text. When used in a logic field, the macro may include any logic element valid in that logic field.

Suppose five different locations use the same lines of code. Instead of entering the same logic five times in five different fields, reference the following macro by typing the macro ID, Mac1, in the operation logic of the machine:



```
INC Var1
IF Var1<10 THEN
   BEGIN
      WAIT 10
      USE Res1 FOR 30
   END
ELSE
   BEGIN
      USE Res1 FOR 20
      USE Res2 FOR 10
   END
```

Every time the macro is referenced, the logic is executed. Macros can also be nested within other macros. This means that a macro can consist of one or more other macros. Consider the following Macro edit table:



ID	Text...	RTI
The	CREATE	None
Race	2	None
For_Quality	AS	None
Has	EntB	None
No	TAKE	None
Finish	1	None
Line	Res1	None
Favorite_Quote	The Race For_Quality Has No Finish Line	None

The macro, Favorite_Quote references other macros, such as Race and Finish. Note that some of the other macros, such as For_Quality, are only portions of a complete line of code. Although the macro is valid, it will not compile as a part of macro logic because the create statement requires an expression and an entity name. The line Favorite_Quote in a logic field would be interpreted as the following line, Create 2 As EntB Take 1 Res1.

Please note

Macro notes:

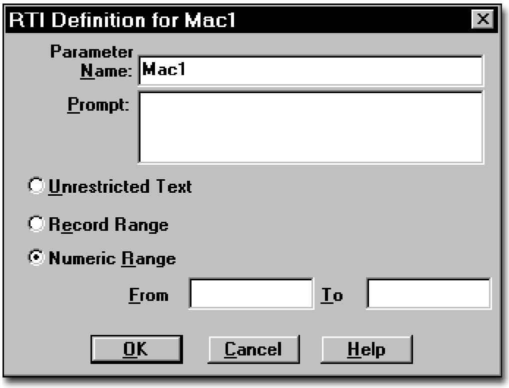
1. A macro may be used only when the elements contained in the macro are appropriate to the context from which it was called. This restriction means that the macro in the previous example is only valid in operation logic.

2. Because a macro simply substitutes some text for its name, if a macro represents a statement block, then it should contain a BEGIN at the beginning of the block, and an END at the end of the block. This technique is especially important when using a macro immediately after a control statement, such as IF...THEN or WHILE...DO. For more information, see "Statement Blocks" on page 436.

Run-Time Interface

Defining a run-time interface (RTI) for a macro allows the user to easily change simulation/model parameters before the simulation starts. It also provides an experimental framework for defining multiple scenarios to be run in a batch (see "Scenarios" on page 353). An RTI for a macro allows a macro's text to be changed by the user whenever a simulation run begins. Since macros are allowed in any expression, this gives the user flexibility to edit most model parameters every time a simulation starts without having to directly edit the model data.

The key difference between a macro with an RTI and one without is that when a simulation begins, a macro defined with an RTI provides a menu that allows users to change only the macros you want them to change. An RTI allows you to request a variety of information to substitute for a macro; from simple values (e.g., the initial value of a location's capacity) to complex text (e.g., a line of logic). You may create RTI parameters using the dialog box below, accessed through the macros dialog.



RTI Definition for Mac1

Parameter Name: Mac1

Prompt:

○ Unrestricted Text

○ Record Range

● Numeric Range

From □ To □

Parameter Name This text will identify the parameter represented by this macro. It should consist of text that clearly describes the parameter to be changed, for example, Operation Time. The macro name and the parameter name can be different. This provides more flexibility and allows you to view a more descriptive parameter name when defining scenarios.

Prompt This text will appear if the user decides to change the parameter. You should use it to further specify the information to be entered, for example, "Please enter the amount of time the simulation should run."

Unrestricted Text This option allows the user to enter any text, such as the distribution U(8,2). Note that any text that the user enters will be substituted for the macro name in the model. Therefore, the text the user enters must be syntactically correct and valid anywhere the macro name appears.

Record Range Allows you to enable an arrival or shift record from a range of records. This allows you to test a variety of shift and arrival combinations to find the combination that works best with your model.

Numeric Range This specifies the lower and upper limits for the parameter if the type is numeric.

How to define an RTI for a macro:

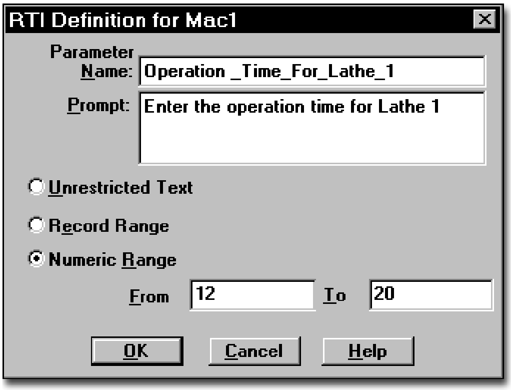
1. Select **Macros** from the **Build** menu.
2. Type the macro name and click on the **Options** button.
3. Choose **Define** from the **RTI** submenu.
4. Define the Parameter Name and enter the prompt (optional).
5. Select the parameter type: **Unrestricted Text**, **Record Range**, or **Numeric Range**.
6. If the parameter is numeric, enter the lower value in the From box and the upper value in the To box.
7. Click **OK**.
8. Enter the default text or numeric value in the Macro Text field.
9. Use the macro ID in the model (e.g., in operation or resource usage time).

Please note

When using a record range, be sure to group all arrival and shift records. This will allow you to select which series of records to include in the macro. Note also that when you define an arrival or shift RTI, ProModel adds "ARRIVAL_" or "SHIFT_" to the name to help you identify the macro more easily.

Run-Time Interface Example

Suppose you build a factory model and determine the first lathe, Lathe_1, is a bottleneck. The model results indicate the throughput is lower than expected. You decide to perform several what-if scenarios with the model by changing the operation time of Lathe_1. Instead of changing the operation time at Lathe_1 within the Process edit table, it is easier to define a macro with an RTI. This technique allows the model user to easily see the effect of installing a faster lathe without ever editing the model itself. The following example represents the dialog used to define the RTI for the macro where the operation time is a numeric value between 12 and 20:



Parameter Name: Operation _Time_For_Lathe_1

Prompt: Enter the operation time for Lathe 1

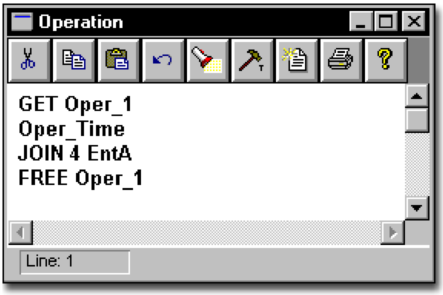
○ Unrestricted Text

○ Record Range

● Numeric Range

From 12 To 20

After defining the RTI for the macro, substitute the macro, Oper_Time, for the operation time in the operation logic in the Process edit table for Lathe_1 as shown below:



```
GET Oper_1
Oper_Time
JOIN 4 EntA
FREE Oper_1
```

You are now able to change the operation time at Lathe_1 using the Model Parameters option in the Simulation menu. For more information on changing model parameters and defining scenarios, see “Model Parameters & Scenarios” on page 352.

Please note

For more information concerning the differences between macros and subroutines, see “Macros” on page 241 and “Subroutines” on page 246.

Resource Grouping

Resource grouping allows you to define specific groups of resources rather than define each unit separately. For example, suppose you need a specific technician to perform an operation. If the technician is not available, you may use either another technician or one of two qualified operators to perform the operation. Rather than define each qualified operator as a separate resource, you may define a macro that includes them.

How to define a resource group

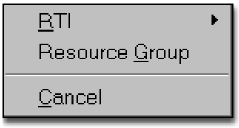
1. Select **Macros** from the more elements section of **Build** menu.

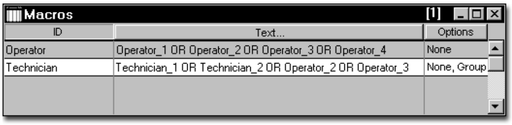
2. Define a macro ID and enter a list of all resources you wish to include as part of the resource group.

Please note

When you create a list of resources, separate each resource using AND or OR (e.g., Tech_1 AND Tech_2 OR Tech_3 AND Tech_4).

3. Click on the **Options** button and select **Resource Group** from the submenu.





ID	Text...	Options
Operator	Operator_1 OR Operator_2 OR Operator_3 OR Operator_4	None
Technician	Technician_1 OR Technician_2 OR Operator_2 OR Operator_3	None, Group

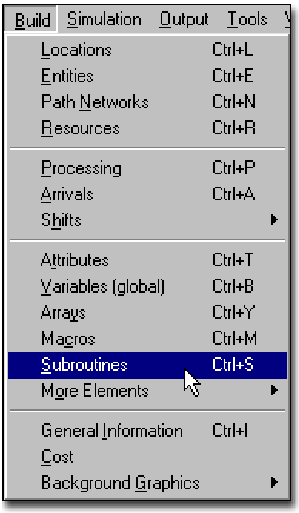
At the end of the simulation, ProModel creates a statistical report containing information collected for each resource included in the resource group, as well as the entire group. This will allow you to track individual, as well as group performance. For information on statistics and how to graph the results, see “Reports and Graphs” on page 373.

Subroutines

A subroutine is a user-defined command that can be called to perform a block of logic and optionally return a value. Subroutines may have parameters (local variables) which act as variables local to the subroutine and that take on the values of arguments (i.e., numeric expressions) passed to the subroutine.

ProModel handles subroutines in three ways. First, a subroutine may be processed by the calling logic as though the subroutine is part of the calling logic. This way is the most commonly used, and is done by simply referencing the subroutine by name in some logic or expression. Second, a subroutine may be processed independently of the calling logic so the calling logic continues without waiting for the subroutine to finish. This method requires an ACTIVATE statement followed by the name of the subroutine (see “Activate” on page 441), or you may use the Interact Menu (see “Run-Time Interact Menu” on page 368). Third, ProModel allows subroutines written in an external programming language to be called through the XSUB() function.

Subroutines are defined in the Subroutines Editor which is accessed from the Build Menu.

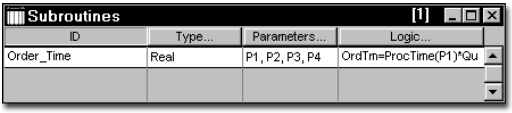


How to create and edit subroutines:

1. Select **Subroutines** from the **Build** menu.

Subroutine Editor

The Subroutines edit table consists of several fields which identify the components of a subroutine. Each of these fields is described below.


Subroutines [1]

ID	Type...	Parameters...	Logic...
Order_Time	Real	P1, P2, P3, P4	OrdTm=ProcTime(P1)*Qu

ID A name that identifies the subroutine.

Type The type of numeric value returned by the subroutine can be Real, Integer, None, or Interactive. Use Real and Integer if the subroutine returns a number and None when no return value is expected, as is often the case in initialization or termination logic. Subroutines of type Interactive are identical to subroutines of type None, except that interactive subroutines are also accessible for activation by the user through the run-time menu. Interactive subroutines are displayed in the Interact menu during runtime. For more information on Interactive subroutines, see the discussion later in this section.

Parameters Arguments passed to the subroutine get assigned to local variables called parameters. Items passed to a subroutine as arguments can have different names than the parameters that receive them. Parameters can be real or integer. The first parameter receives the first argument, the second parameter receives the second argument, and so on.

Logic One or more statements to be executed whenever the subroutine is called. Statements in subroutines must be valid in the logic that calls the subroutine. Subroutine logic may contain a RETURN statement with a value to be returned from the subroutine. (See “Return” on page 549 for the correct syntax and an example of this statement.)

Please note

Subroutine editor notes:

1. If the subroutine is of type Integer and the return value is a real number, the return value will be truncated unless the ROUND() function is used (e.g., RETURN ROUND(<numeric expression>)).

2. If you do not want a stand-alone subroutine referenced in operation logic to be treated as an implicit wait statement, define the subroutine as type None.

3. When using the ACTIVATE statement to call a subroutine, the calling logic continues without waiting for the called subroutine to finish. Therefore, independent subroutines can run in parallel with the logic that called them.

4. Independent subroutines called with the ACTIVATE statement cannot contain entity-specific or location-specific system functions.

Subroutine Format

A subroutine may be named any unique, valid name. The general format for calling a subroutine is as follows:

```
SubroutineName(arg1, arg2,....,argn)
```

Subroutine call

```
GetOpTime(3,7)
DoInitialization()
```

Please note

Subroutine format notes:

1. If no arguments are specified, open and closed parentheses are still required.

2. Statements in subroutines must be valid in the logic that called the subroutine. For example, if a subroutine is called from the operation logic, the subroutine may contain only those statements valid in the operation logic. Subroutines called from an ACTIVATE statement or from the Interact Menu at run-time can have any general logic statements, including WAIT.

3. A subroutine may be used in any logic field. In addition, a subroutine may be used in any expression field, provided that the RETURN statement is used to return a value to the expression field.

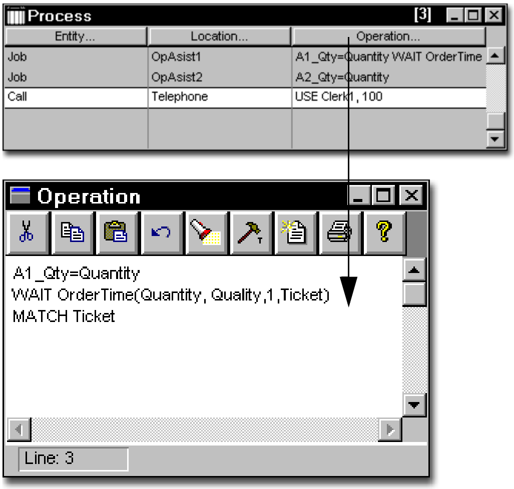
Expression fields include the Qty Each column of the Arrivals edit table and the routing rule for processing.

4. If a subroutine does not return an expression with the RETURN statement, a value of zero will be returned for subroutines of type Real and Integer. No value will be returned for a subroutine of type None or Interactive.

Subroutine Example

Suppose that you build a Copy Center model and it is necessary to define the processing time at the operator assisted machines as a function of two parameters: Order Quantity and Quality Level. In addition, you would also like to write the processing times at each copy machine to a file called Report.Dat.

Because the operation logic is identical at both machines, a subroutine may be used to execute the group of statements required. Notice in the Process Logic pictured next that Subroutine OrderTime is called, and each order's Quantity, Quality, and Ticket attributes are passed as arguments. The third argument is simply a constant value, 1 or 2, which represents the location from which the subroutine was called (i.e., 1 if called from OpAsist1 or 2 if called from OpAsist2). The operation logic, which calls the subroutine, and routing are shown below. The subroutine itself is shown later in this section.

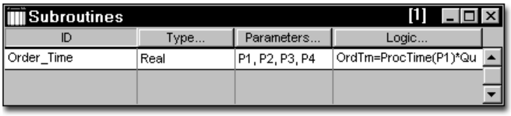


Entity...	Location...	Operation...
Job	OpAsist1	A1_Qty=Quantity WAIT OrderTime
Job	OpAsist2	A2_Qty=Quantity
Call	Telephone	USE Clerk1, 100

```
A1_Qty=Quantity
WAIT OrderTime(Quantity, Quality,1,Ticket)
MATCH Ticket
```

Subroutines Edit Table

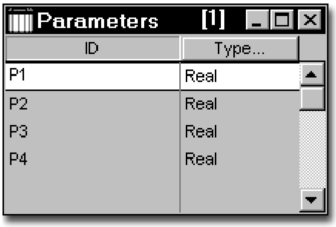
The Subroutine edit table lists the name of the subroutine, the return type, the parameters to be passed to the subroutine, and the logic. Notice that the subroutine ID corresponds to the subroutine name called in the processing logic above and that the subroutine has four parameters (P1, P2, P3 and P4) corresponding to the four arguments passed to the subroutine. Also note that the subroutine is of type real since the return value will be a real valued processing time.



ID	Type...	Parameters...	Logic...
Order_Time	Real	P1, P2, P3, P4	OrdTm=ProcTime(P1)*Qu

Subroutine Parameters

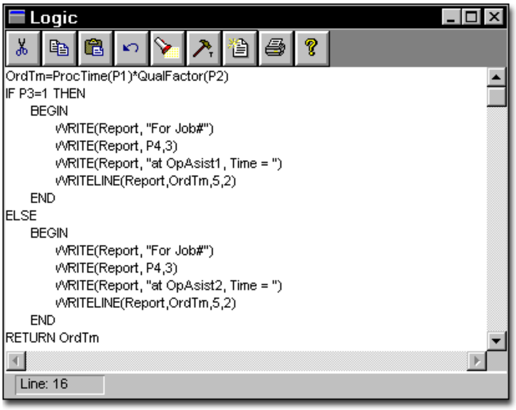
The subroutine parameters, M (for mean) and SD (for standard deviation), are defined by clicking on the Parameters heading button. These values are unique to each inspection location, and are passed to the subroutine as parameters of the normally distributed inspection time.



ID	Type...
P1	Real
P2	Real
P3	Real
P4	Real

Subroutine Logic

The final step in defining the subroutine is to specify the logic to be processed when the subroutine is called. In this example, the logic should include a processing time, a procedure to write the values of the attributes to a file, and a routing decision based on the values of the two attributes. The logic window is accessed by clicking on the Logic heading button.



```
OrdTm=ProcTime(P1)*QualFactor(P2)
IF P3=1 THEN
    BEGIN
        WRITE(Report, "For Job#")
        WRITE(Report, P4,3)
        WRITE(Report, "at OpAsist1, Time = ")
        WRITELINE(Report,OrdTm,5,2)
    END
ELSE
    BEGIN
        WRITE(Report, "For Job#")
        WRITE(Report, P4,3)
        WRITE(Report, "at OpAsist2, Time = ")
        WRITELINE(Report,OrdTm,5,2)
    END
RETURN OrdTm
```

The first line of the logic calculates the order time as a function of the Quantity and Quality attributes passed as parameters. These parameters are used as arguments in two user defined functions, ProcTime() and QualFactor(). Function ProcTime() returns a time value based on the number of copies in the order, while function QualFactor() returns a scale factor depending upon the quality level desired. Operation time is determined by simply multiplying the process time by the scale factor.

Once the order time has been determined, this value is written to the file Report.Dat. Included with the operation time is the job number and the location at which the job was processed.

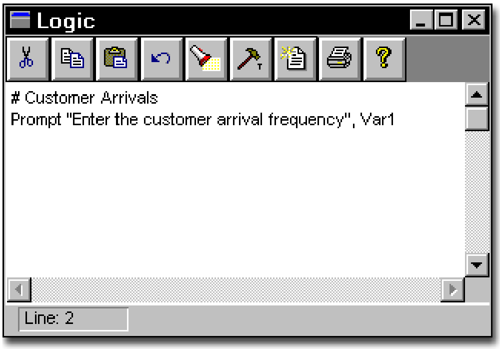
The last line of the logic returns the order time value to the processing logic.

Interactive Subroutines

Interactive subroutines are subroutines activated by the user anytime during run-time by choosing the subroutine from the Interact menu. The name appearing in the Interact menu is either the subroutine name or if a string is entered as a comment statement at the beginning of the subroutine logic, the string is used as the name (e.g., # “Arrival Frequency”). Interactive subroutines allow the user to interact with the simulation during run-time. Subroutines are defined as type Interactive in the Subroutine edit table. Normally, subroutines are activated by entities. However, interactive subroutines can be user-activated in addition to being entity-activated. Interactive subroutines can be used for:

- changing model parameters during a simulation run
- changing routings
- calculating and reporting user-defined statistics

Suppose you want to interactively change the arrival frequency for a certain entity, customers, during run time. Define a variable, Var1, and assign it an initial value to be used for the initial arrival frequency. Enter Var1 in the arrival frequency field for the entity customers. Create a subroutine of type Interactive and enter the following logic:



```
# Customer Arrivals
Prompt "Enter the customer arrival frequency", Var1
```

During run-time, you can then change the arrival frequency for the customers by choosing the Customer Arrivals from the Interact menu (see “Run-Time Interact Menu” on page 368).

Please note

Interactive subroutines may also be called from any logic or expression where no return value is required. See “Statements and Functions” on page 439 for more information.

External Subroutines

There may be some cases where you need to perform actions ProModel is not capable of doing. You may need extended capabilities with more sophisticated commands. ProModel allows you to interface with external subroutines located in thirty-two bit Windows DLL files you have created. This feature could be useful for doing sophisticated file I/O, performing statistical analysis, making your simulation interactive, or helping with other simulation needs.

Because of the intricacies of the Windows development environment, you must have a sound knowledge of your external programming language (C, C++, Pascal) to use external subroutines. In addition, you must also have a good Windows platform knowledge, specifically with respect to creating DLLs in your language. Because it is a 32-bit program, ProModel can load and call only 32-bit DLLs, and requires that you use a 32-bit Windows compiler.

For more information about this feature, you can load, study and run XSUB.MOD in the reference model directory (also see “Xsub()” on page 583). This model uses XSUB.DLL, found in the MODELS directory. The source code and make files for XSUB.DLL (XSUB.CPP, XSUB.MAK, XSUB.IDE) are also included in the MODELS directory. Some general explanation is contained in the comments of this source code.

Due to the complexities of Windows programming and the variety of uses for this advanced feature, PROMODEL Corporation can only provide minimal support for this feature. Many questions regarding Windows programming and other programming languages cannot be handled through our customer support department. Please consult your language programming manuals, language customer service centers, Microsoft, and other resources to resolve these types of problems.

Subroutines vs. Macros

Although subroutines and macros work similarly, they have subtle differences. Any logic may use both macros and subroutines. The main difference is in the way they are used. Only subroutines can be used when you need to pass arguments, get a return value, or activate the independent execution of logic. Only macros can be used when defining run-time interface parameters.

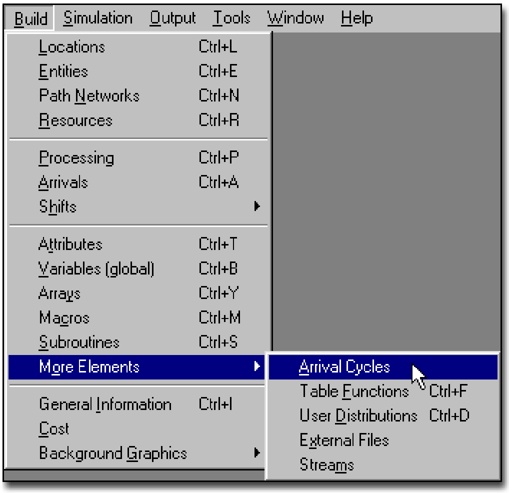
Macros may be used in any expression field, but the macro may only contain an expression (i.e., Entries(LOC1), U(5,1)). When a macro is used in a logic field, the macro may include any logic element valid in that logic field.

Subroutines may also be used in an expression field provided that the RETURN statement is used to return a value to the expression field. When a subroutine is used to represent one or more logic statements, the subroutine may only include statements valid for the particular context.

Arrival Cycles

An arrival cycle is a pattern of individual arrivals which occurs over a certain time period. Some examples of arrival cycles that exhibit a pattern are the arrival of customers to a store and the arrival of delivery trucks to a truck dock. At the beginning of the day, arrivals may be sparse; but as the day progresses, they build up to one or more peak periods and then taper off. While the total quantity that arrives during a given cycle may vary, the pattern or distribution of arrivals for each cycle is assumed to be the same.

Arrival Cycles are defined in the Arrival Cycles edit table, accessed through the Build Menu.



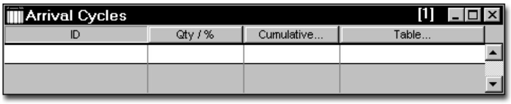
How to edit arrival cycles:

1. Choose **More Elements** from the **Build** Menu.

2. Choose **Arrival Cycles**.

Arrival Cycles Edit Table

Arrival cycles are defined by entering the proper data into the Cycle edit table. The fields of the Cycle edit table are explained below.



ID The cycle name.

Qty / % Select either Percent or Quantity as the basis for the total number of arrivals per cycle occurrence.

Cumulative... Select Yes to specify the % or Qty values in a cumulative format. Select No to specify these fields in a non-cumulative format.

Table... Click on this button (or double click in this field) to open an edit table for specifying the cycle parameters.

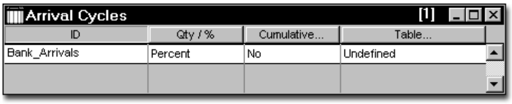
Arrival Cycles Example

Suppose we are modeling the operations of a bank (or any service or manufacturing system) and we need to specify a pattern for customer arrivals. From past data, we know that customers arrive throughout the day (9:00 AM to 5:00 PM) according to the following approximate percentages.

From	Before	Percent
9:00 AM	10:30 AM	0
10:30 AM	11:30 AM	10
11:30 AM	1:00 PM	25
1:00 PM	4:00 PM	55
4:00 PM	5:00 PM	70

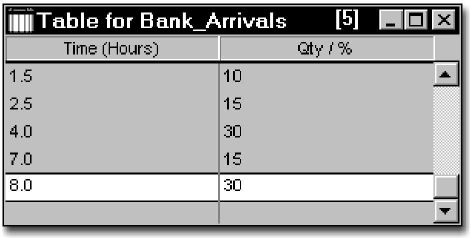
Defining the Arrival Cycle

The arrival cycle will be named Bank_Arrivals in the Arrival Cycles edit table. Because the data is expressed in terms of percentages, we select Percent as the basis for the cycle. Also, the percentage values are not cumulative so we specify No in the Cumulative field.



ID	Qty / %	Cumulative...	Table...
Bank_Arrivals	Percent	No	Undefined

Next, we click on the Table heading button to open another edit table for entering the cycle data.



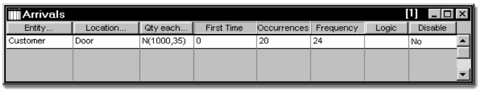
Time (Hours)	Qty / %
1.5	10
2.5	15
4.0	30
7.0	15
8.0	30

In this example, even though the percentages of customers that have arrived are not cumulative, the time is always cumulative. Therefore, the table reads as follows: ten percent of the daily customers arrive in the first 1.5 hours, fifteen percent of the daily customers arrive between hour 1.5 and 2.5, and so on. The arrivals are randomly distributed according to a Uniform distribution during the time interval in which they arrive.

The arrival cycle is now defined and can be assigned to an arrivals record in the Arrivals edit table.

Assigning Arrivals to the Arrival Cycle

After an arrival cycle has been defined, assign an arrival record to the arrival cycle in the Arrivals edit table for the arriving customers. The arrival record for the bank example appears below. (See “Arrivals” on page 163 for more information.)

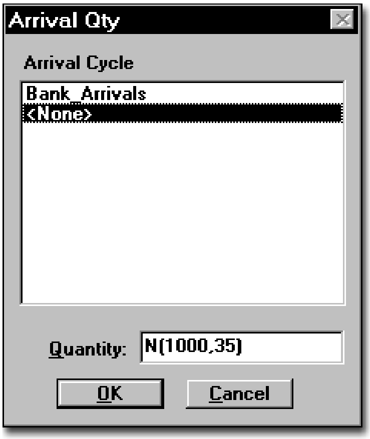


Entity...	Location...	Qty each...	First Time	Occurrences	Frequency	Logic	Disable
Customer	Door	N(1000,35)	0	20	24		No

The total number of customers per day is normally distributed with a mean of 1000 and a standard deviation of 35. The number of occurrences of the cycle is 20, representing 20 working days (1 month) of time. The frequency in this case refers to the period of the cycle or the time between the start of one cycle and the start of the next cycle, which is every 24 hours. Make sure the arrival frequency is defined with the same time unit as the arrival cycle.

Without an arrival cycle, all the Customers for a single day would arrive at the start of the simulation. An arrival cycle will divide the quantity specified in the Qty each... field into various sized groups arriving throughout the day.

To assign the arrival cycle to the arrival record, click on the Qty each... heading button to open a dialog with the names of all defined cycles.



Bank_Arrivals
<None>

Quantity: N(1000,35)

Click on the entry Bank_Arrivals and select OK. The Qty each... field now includes the cycle.

Cumulative Cycle Tables

In the previous example, percentages were expressed non cumulatively. This same data could have been expressed cumulatively as follows:

Before	Percent
9:00 am	0
10:30 am	10
11:30 am	25
1:00 pm	55
4:00 pm	70
5:00 pm	100

The data is now expressed cumulatively and could be entered in the cycle table as follows.

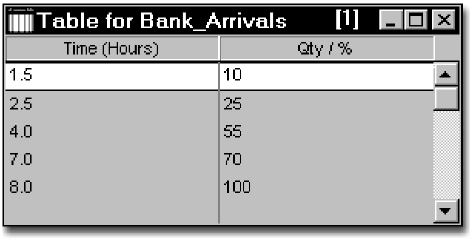


Table for Bank_Arrivals [1]

Time (Hours)	Qty / %
1.5	10
2.5	25
4.0	55
7.0	70
8.0	100

To specify cycles in cumulative form, simply choose Yes in the Cumulative... field of the Arrival Cycles edit table.

Please note

Time values remain cumulative regardless of the form of the percentages.

Arrival Cycles by Quantity

The previous example was based on the assumption that a certain percentage of arrivals came within a specified time interval. An alternate method of specifying an arrival cycle is to specify the number of arrivals to arrive within each time interval.

Example 1

Suppose that in the bank example we knew that for each cycle period, the number of customers to arrive during each time interval within the cycle period is as follows:

From	Before	Number
9:00 AM	10:30 AM	100
10:30 AM	11:30 AM	150
11:30 AM	1:00 PM	300
1:00 PM	4:00 PM	150
4:00 PM	5:00 PM	300

With the data in this format, we specify the Arrival cycle by choosing Qty in the “Qty/Percent” field and complete the cycle table as follows:

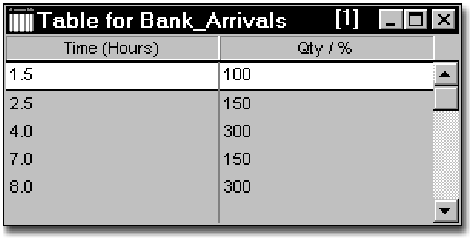


Table for Bank_Arrivals [1]

Time (Hours)	Qty / %
1.5	100
2.5	150
4.0	300
7.0	150
8.0	300

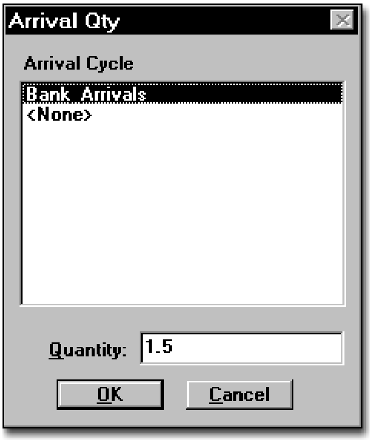
This table could also be specified cumulatively by choosing Yes in the Cumulative... field and entering the quantity values in a cumulative format.

Please note

When specifying Arrival cycles by quantity, the value entered in the “Quantity each” field of the Arrivals edit table changes meaning. Instead of the total arrivals per cycle, it represents a factor by which all entries in the cycle table will be multiplied. This field may be any valid expression which evaluates to a number. This allows the *same arrival cycle to be entered for more than one arrival record that has a different factor applied to it. For example, you might want to define a different factor to a customer arrival pattern depending on the day of the week. In this case, define an arrival record for each day with a frequency of one week.*

Example 2

Suppose we wish to see the effect on the Bank example if the number of arrivals is increased by 50%. The relative quantities per time interval remain the same but now 50% more customers arrive each day. Using the data from the previous example, we enter the same values in the Arrival cycle quantity fields, but specify a value of 1.5 in the quantity field of the Arrival edit table.



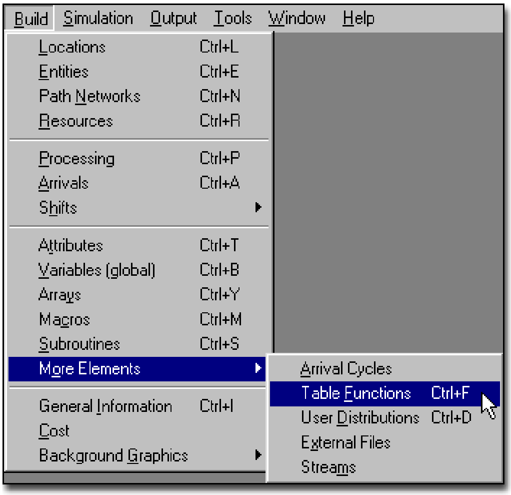
Quantity: 1.5

The result is an arrival schedule with the following parameters.

From	Before	Quantity
9:00 AM	10:30 AM	150
10:30 AM	11:30 AM	225
11:30 AM	1:00 PM	450
1:00 PM	4:00 PM	225
4:00 PM	5:00 PM	450

Table Functions

Table functions provide an easy and convenient way to retrieve a value based on an argument (i.e., some other value) that is passed to the table. Table functions specify a relationship between an independent value and a dependent value. All table functions are defined in the Table Functions editor which is accessed from the Build menu.



How to access the function table editor:

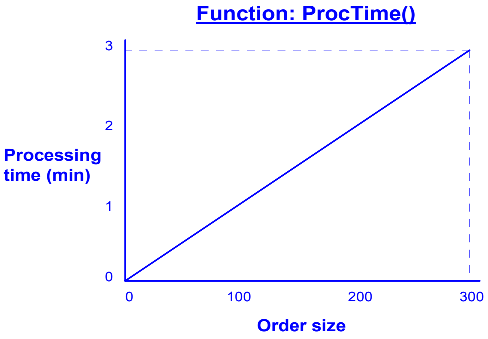
1. Select **More Elements** from the **Build** menu.
2. Select **Table Functions**.

Table Functions Editor

Table functions are defined by the user and return a dependent (or look-up) value based on the independent (or reference) value passed as the function argument. Independent values must be entered in ascending order. If the independent value passed to a table function falls between two independent values, a dependent value for the unspecified reference value is calculated by linear interpolation. The following two examples show how to specify a linear function (where only two reference values are needed to define the entire function) and a nonlinear function (where more than two reference values need to be specified).

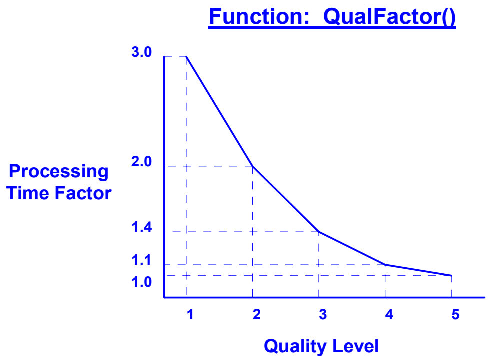
Example 1

The example below shows a relationship that exists between the time required to process an order and the number of entities in the order. As the number of entities increases, the time required to process the order also increases. The relationship in this case is linear, meaning the processing time is directly proportional to the number of copies in the order. Because of the linear relationship, only the two endpoints need to be entered in the function table. (The function table for this example is given in the discussion on the Table Function editor.)



Example 2

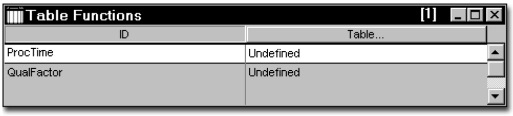
In this example the relationship between the independent value and the dependent value is nonlinear and inversely proportional. In addition, interpolation is required to determine the dependent value if the independent value passed to the function lies between the independent values given explicitly in the table function.



In the example above the dependent value represents a factor which is to be multiplied by the processing time required to complete an order. Each order is assigned a number, 1 through 5, according to the level of quality desired. In this case, a lower number represents higher quality. (The function table for this example is listed in the following explanation of the Function Table Editor.)

Table Function Edit Table

The Function Table Editor is where all function tables are created and edited. The fields of the Function Table editor are defined below.

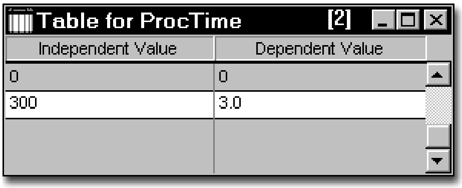


ID	Table...
ProcTime	Undefined
QualFactor	Undefined

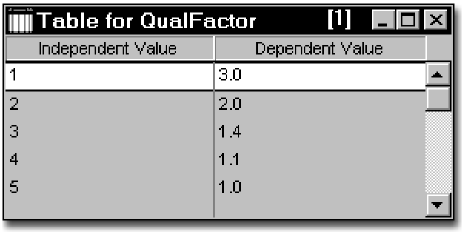
ID The name of the function table.

Table... Click on this heading button to open a table for defining the independent and dependent values of the function.

The tables for the two example functions are given below.



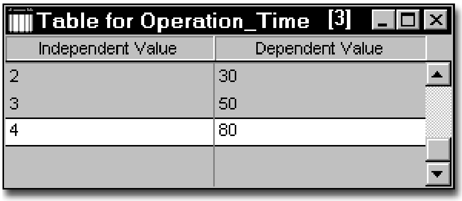
Independent Value	Dependent Value
0	0
300	3.0



Independent Value	Dependent Value
1	3.0
2	2.0
3	1.4
4	1.1
5	1.0

The independent and dependent values allow any general expression such as numbers, variables, and math functions. The fields are evaluated only at translation, and cannot vary during the simulation.

When calling a user-defined table function, if the independent value is out of range, then the table function will return a zero for the dependent value. Consider the following function table, Operation_Time:



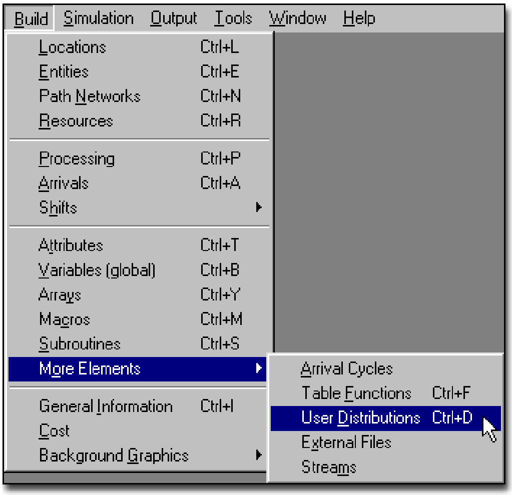
Independent Value	Dependent Value
2	30
3	50
4	80

If the function were called with the command "Operation_Time(5)," the independent value passed to the table function Operation_Time would be five. But five is beyond the limits of the table, so the dependent value returned will be zero. Likewise, if the independent value is 1, the dependent value returned will be zero. However, if 2.7 is entered as the independent value, Pro-

Model will interpolate and return a value between 30 and 50.

User Defined Distributions

Occasionally, none of ProModel's built-in distributions can adequately represent a data set. In these cases, the user may define a User Distribution to represent the data set. User Distributions specify the parameters of user-defined (empirical), discrete, or continuous probability distributions.

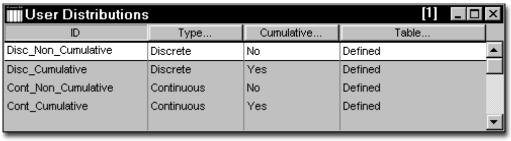


How to create and edit user distributions:

1. Choose **More Elements** from the **Build** menu.
2. Choose **User Distributions**.

User Distribution Edit Table

A user-defined distribution is a table of empirically gathered data. User distributions may be either continuous or discrete, and may be cumulative or non-cumulative (more information concerning these options is found later in this section). The data is entered into the User Distribution edit table. The User Distribution edit table's fields are described below.



ID	Type...	Cumulative...	Table...
Disc_Non_Cumulative	Discrete	No	Defined
Disc_Cumulative	Discrete	Yes	Defined
Cont_Non_Cumulative	Continuous	No	Defined
Cont_Cumulative	Continuous	Yes	Defined

ID The name of the distribution. When referencing distribution tables (in the operation logic, for example) the open and closed parentheses after the distribution name must be used, such as Dist1(), OpTime().

Type... Discrete or Continuous depending on the number of possible outcomes.

Cumulative... Yes or No depending on whether the distribution is to be specified in cumulative or non-cumulative format.

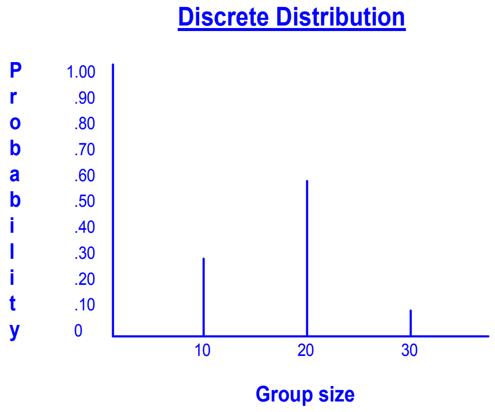
Table... Click on this button (or double click in the field) to open an edit table for defining the parameters of the distribution. Once a distribution has been defined, the field changes from "Undefined" to "Defined."

The combination of Discrete and Continuous distributions, along with the ability to express either in cumulative or non-cumulative terms, creates four possible formats for specifying distributions. The remainder of this section gives examples and procedures for specifying each of these distribution types.

Discrete Distributions

Discrete distributions are characterized by a finite set of outcomes, together with the probability of obtaining each outcome. In the following example, there are three possible outcomes for the group size: 30% of the time the group size will be

10, 60% of the time the group size will be 20, and 10% of the time the group size will be 30.



One way to represent a discrete distribution is by its probability mass function, listing the possible outcomes together with the probability of observing each outcome. A probability mass function for the example above could be expressed as follows (with G representing the group size).

G	10	20	30
P(G)	.30	.60	.10

An alternate way to represent a distribution is through a cumulative distribution function, listing each possible outcome together with the probability that the observed outcome will be less than or equal to the specified outcome. A cumulative distribution function for the example above could be expressed as follows.

G	10	20	30
P(G)	.30	.90	1.0

In the next example, the number of parts are grouped into a batch according to a user distribution.

Process Table

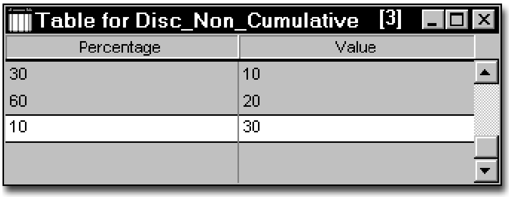
	Location	Operation (min)
EntA	Loc1	**GROUP Dist() AS Batch**
Batch	Loc1	WAIT 10 min

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Batch	Loc2	FIRST 1	MOVE FOR 5

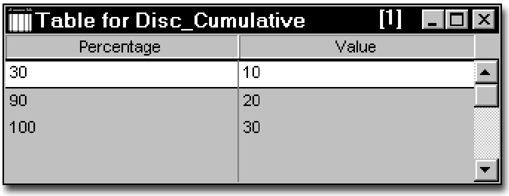
ProModel provides the flexibility to specify discrete distributions according to a probability mass function or a cumulative distribution function. Select Yes or No in the Cumulative field of the Distribution edit table and fill in the table according to the probability mass function or the cumulative distribution function. The following tables show the discrete distribution example defined in both formats.

Discrete (probability mass function)



Percentage	Value
30	10
60	20
10	30

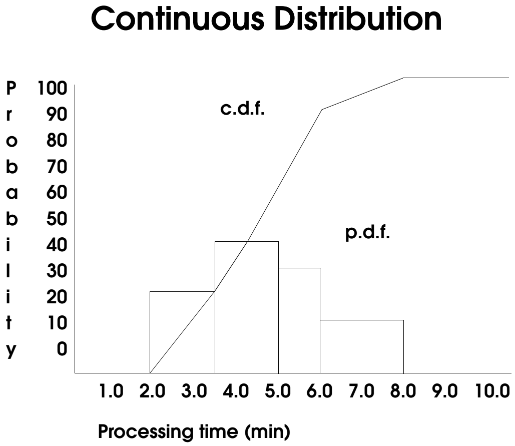
Discrete (cumulative distribution function)



Percentage	Value
30	10
90	20
100	30

Continuous Distributions

Continuous distributions are characterized by an infinite number of possible outcomes, together with the probability of observing a range of these outcomes. In the following example, there are an infinite number of possible operation times between the values 2.0 minutes and 8.0 minutes. Twenty percent of the time the operation will take from 2.0 to 3.5 minutes, 40% of the time the operation will take from 3.5 to 5.0 minutes, 30% of the time the operation will take from 5.0 to 6.0 minutes, and 10% of the time the operation will take from 6.0 minutes to 8.0 minutes.



Continuous Distribution

As with a discrete distribution, a continuous distribution can be defined in two ways. A probability density function lists each range of values along with the probability that an observed value will fall within that range. Each of the values within the range has an equal chance of being observed, hence the piece-wise linearity of the c.d.f. within each range of values. A probability density function for the example above is expressed as follows.

P(0.0 <= X < 2.0)	**= 0.00**
P(2.0 <= X < 3.5)	**= 0.20**
P(3.5 <= X < 5.0)	**= 0.40**
P(5.0 <= X < 6.0)	**= 0.30**
P(6.0 <= X <= 8.0)	**= 0.10**
P(8.0 < X)	**= 0.00**

The following table represents the p.d.f. for this example.

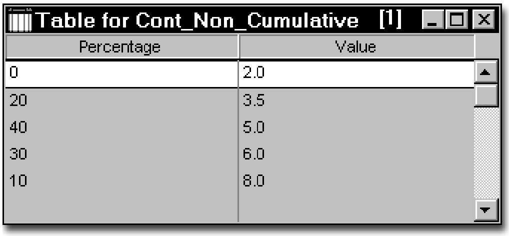


Table for Cont_Non_Cumulative [1]

Percentage	Value
0	2.0
20	3.5
40	5.0
30	6.0
10	8.0

As with a discrete distribution, a cumulative distribution function for a continuous distribution specifies the probability that an observed value will be less than or equal to a specified value. A c.d.f. for the example distribution is as follows (where x represents the return value).

x	**2.0**	**3.5**	**5.0**	**6.0**	**8.0**
P(X <= x)	**0**	**.20**	**.60**	**.90**	**1.0**

The following table represents this c.d.f.

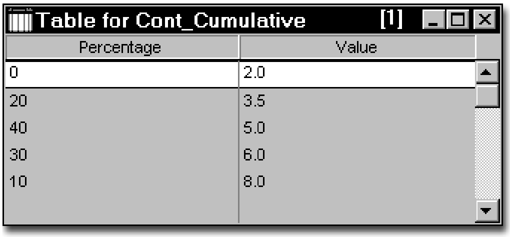
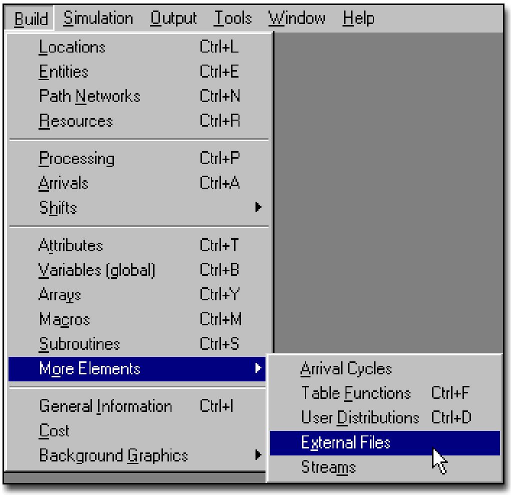


Table for Cont_Cumulative [1]

Percentage	Value
0	2.0
20	3.5
40	5.0
30	6.0
10	8.0

External Files

External files may be used during the simulation to read data into the simulation or write data as output from the simulation. Files can also be used to specify such things as operation times, arrival schedules, shift schedules, and external subroutines. All external files used with a model must be listed in the External Files Editor which is accessed from the Build menu.

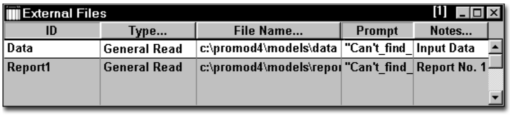


How to define external files:

1. Select **More Elements** from the **Build** menu.
2. Select **External Files**.

External Files Editor

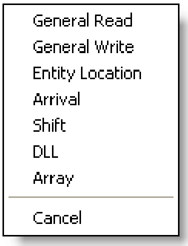
The External Files Editor consists of an edit table with fields specifying the external files to be used during the simulation. Each of these fields is defined below.



ID	Type...	File Name...	Prompt	Notes...
Data	General Read	c:\promod4\models\data	"Can't_find_	Input Data
Report1	General Read	c:\promod4\models\repor	"Can't_find_	Report No. 1

ID An alias to be used in the model for referencing the file. Note that this ID does not have to be the same as the file name.

Type Click on this heading button to display the following menu. The six file types are discussed in the remainder of this section.



File Name... The actual file name, including the path. Press the heading button or double click in this field to browse for a specific file.

Prompt A prompt to be displayed at run time in the event that the specified file cannot be opened.

Notes... A general notes field for entering descriptive information about the file. Click the heading button or double click in this field to open a larger window for entering notes.

File Types

External files may be defined as one of several types depending upon the purpose of the file.

General Read File

A General Read file contains numeric values read into a model using a READ statement. Values must be separated by a space, comma, or end of line. Any non-numeric data will be automatically skipped when obtaining the next numeric value (See “Read” on page 537 for correct syntax and examples). For example, if you specify a normal distribution such as N(5,1) in the General Read file, ProModel will not return a numeric value following the distribution. Instead, it will read in the first value, 5, and the next value, 1.

A General Read file must be an ASCII file. Data created in a spreadsheet must be saved as a text file.

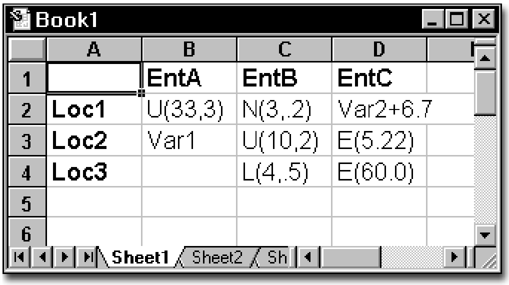
General Write File

A General Write file is used for writing text strings and numeric values using the WRITE and WRITELINE statements. Text strings are enclosed in quotes when written to the file, with commas automatically appended to strings. This enables the files to be read into spreadsheet programs like Excel or Lotus 1-2-3 for custom viewing, editing, and graphing. Write files may also be written to using the XWRITE statement which gives the modeler full control over output and formatting. (See “Write” on page 581, “Write-Line” on page 582, and “Xwrite” on page 585 for correct syntax and examples.)

If you write to an external file during multiple replications or a single, independent run, the data will be appended to the data from the previous replication. However, if the RESET statement is used, the data is overwritten for each replication.

Entity-Location File

An Entity-Location file (or expression file) is a spreadsheet (.wk1 or .xls format only) file containing numeric expressions listed by entity and location name. Entity names should appear across the top row, beginning in column 2, while location names should be entered down the first column, beginning on row 2. A numeric expression for each Entity-Location combination is entered in the cell where the names intersect. An example of a spreadsheet file is shown next.



	A	B	C	D
1		EntA	EntB	EntC
2	Loc1	U(33,3)	N(3,.2)	Var2+6.7
3	Loc2	Var1	U(10,2)	E(5.22)
4	Loc3		L(4,.5)	E(60.0)
5				
6				

To use the value stored in an Entity-Location file as an operation time, call out the file identifier in the operation logic as shown in the following example. (In this example, “SvcTms” is the File ID of the desired Entity-Location file.)

Please note

If the .wk1 or .xls file contains more than one sheet of data, only the first sheet will be read in.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT SvcTms()

Routing Table

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

By specifying SvcTms() with no arguments in the parentheses, a value is returned from the Entity-Location File “SvcTms” for the current entity at the current location, i.e., EntA at Loc1. You may also return the value stored in any other cell of an Entity-Location file by explicitly specifying the entity and location names in the parentheses, e.g., SvcTms(EntB, Loc1) or SvcTms(EntC, Loc2).

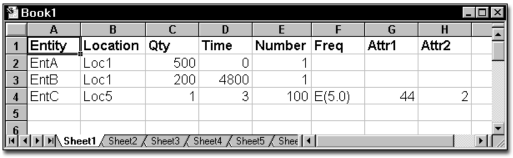
Arrivals File

An Arrivals file is a spreadsheet (.wk1 or .xls format only) file containing arrival information normally specified in the Arrival Editor. One or more arrival files may be defined and referenced in the External Files Editor. Arrival files are automatically read in following the reading of the Arrival Editor data. The column entries must be as follows:

Column	Data
A	Entity name
B	Location name
C	Quantity per arrival
D	Time of first arrival
E	Number of arrivals
F	Frequency of arrivals
G through...	Attribute assignments

Columns A through F may have any heading desired as long as the data is of the proper type. If attributes are to be assigned, columns G and higher should have headings that match the names of the attributes being assigned. The following example illustrates these points.

Example



Book1

	A	B	C	D	E	F	G	H
1	Entity	Location	Qty	Time	Number	Freq	Attr1	Attr2
2	EntA	Loc1	500	0	1			
3	EntB	Loc1	200	4800	1			
4	EntC	Loc5	1	3	100	E(5.0)	44	2
5								
6								

The values in the spreadsheet cells must be a numeric expression as opposed to a formula commonly used in spreadsheets. For example, if cell E4 in the spreadsheet above was actually a formula generating the value 100, the value ProModel generates is zero. ProModel only recognizes expressions for the Qty, Time, Number, and Frequency columns in a spreadsheet.

When defining an External Arrivals File, you do not need to define arrivals in the Arrivals edit table. If several entities are scheduled to arrive at the same time, entities arrive in the system according to the order in which they appear in the arrival list. However, when there is more than one occurrence for the arrival record, the next entity will not arrive until the frequency has elapsed. Meanwhile, other entities listed below the record may be allowed to arrive.

Please note

If the time of the first arrival is zero and there is only one arrival of some quantity, you do not need to complete additional cells. Likewise, if there is only one arrival at a time other than zero, you do not need to fill in additional cells after the Time entry.

Please note

If the .wk1 or .xls file contains more than one sheet of data, only the first sheet will be read in.

Shift File

A shift file record is automatically created in the External Files Editor when you assign a shift to a location or resource. If shifts have been assigned, the name(s) of the shift file(s) will be automatically created in the External Files Editor. If no path is listed for the shift file, ProModel will search in the default models directory.

Please note

Creating a shift file record in the External Files Editor should not be done. It is done automatically through the shift assignment.

DLL File

A DLL file is needed when using external subroutines through the XSUB() function. See “Subroutines” on page 246 for more information.

Excel File

An Excel file is automatically created in the External Files Editor when you assign a shift to an array import file. If the external file has been assigned, the name(s) of the Excel file(s) will be automatically created in the External Files Editor. If no path is listed for the Excel file, ProModel will search in the default models directory.

Other External Files

In addition to allowing the user to define external files, ProModel creates other external files. ProModel automatically creates and/or opens files depending on the specifications in the model. Below is a description of the different files ProModel creates (* indicates files that remain open while the model is running):

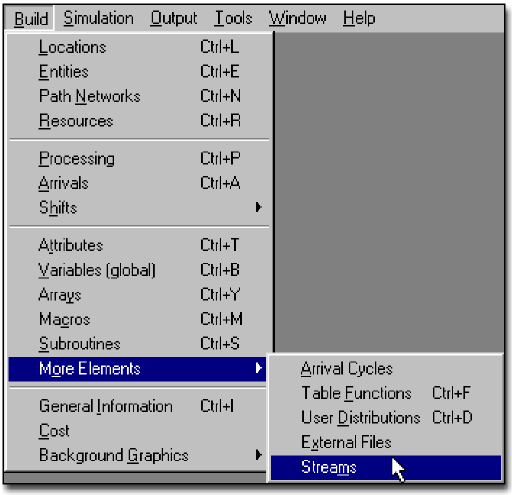
Extension	Type	Description
CSV	ASCII	Export Data (comma delimited)
MOD*	Binary	Model File
RDB	Binary	Output Database (basic statistics)
RDT	Binary	Output Time Series data
RDW	Binary	Output Reports & Graphs saved for a model
SED*	Binary	Seed File used to store seed values for each replication
SFT	Binary	Shift created when defining a shift in the shift editor
TRC*	ASCII	Trace File
GLB	Binary	Graphic Library Files
GBM	Binary	Graphic Bit Map file created when loading a model file.

Open Files

Depending on the model specifications, there may be some occasions where several files need to be open simultaneously to execute the model. There is a feature which allows the user to access up to 255 open files at any time. The [General] section of the **promod.ini** file contains the following statement: OpenFiles=n where n is the number of files between 20 and 255. The default is 40. To change the number of available open files, simply edit the**promod.ini** file such that OpenFiles equals the desired number and then restart ProModel. There is also the option to close files using the CLOSE statement (see *“Close” on page 458*).

Streams

A stream is a sequence of independently cycling, random numbers. Streams are used in conjunction with distributions. Up to 100 streams may be used in a model. A stream generates random numbers between 0 and 1, which in turn are used to sample from selected distributions. By default, all streams use seed value #1 and are not reset between replications if multiple replications are run. To assign a different starting seed value to a stream or to cause the seed value to be reset to the initial seed value between replications, use the Streams Editor. The Streams Editor is accessed from the Build menu as shown below.

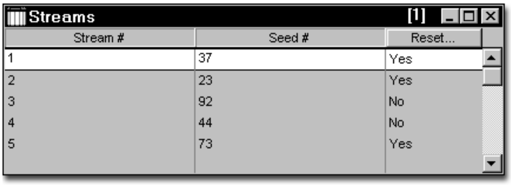


How to access the streams editor:

1. Select **More Elements** from the **Build** menu.
2. Select **Streams** from the submenu.

Streams Edit Table

The Streams Editor consists of an edit table with three fields. The fields of the Streams edit table are explained below.



Stream #	Seed #	Reset...
1	37	Yes
2	23	Yes
3	92	No
4	44	No
5	73	Yes

Stream # The stream number (1 to 100). This number identifies each stream.

Seed # The seed value (1 to 100). Streams having the same seed value generate the same sequence of random numbers. For more information on seed values, see "Using Random Number Streams" on page 266.

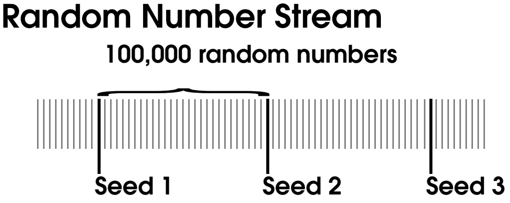
Reset... Set this field to YES if you want the stream to be reset to the initial seed value for each model replication. Set this field to NO if you want the stream to continue where it left off for subsequent replications.

The Streams Editor pictured above shows five defined streams. Each stream has a unique seed value. Streams 1, 2 and 5 will be reset for each replication, while streams 3 and 4 will continue where they left off for subsequent replications.

Using Random Number Streams

One of the most valuable characteristics of simulation is the ability to replicate and isolate probabilistic functions and activities within a system for specific study. In the real world, events tend to occur randomly, according to a certain statistical pattern or distribution. To help you model this randomness, ProModel uses distributions.

When you include a distribution (e.g., Normal, Beta, and Gamma) in your model, ProModel uses a random number generator to produce a set sequence or *stream* of numbers between 0 and 1 (0 <= x < 1) to use in the distribution. Before it can select any numbers, however, ProModel requires an *initial seed value* to identify the point in the stream at which to begin. Once you specify a seed value, ProModel "shifts" the random number selection (in increments of 100,000 numbers) by that number of positions and starts sampling values. Since there is only one random number stream, this will ensure that the selected values do not overlap. ProModel includes 100 seed values, and each seed produces a unique set of random numbers. If you do not specify an initial seed value, ProModel will use the stream number as the seed value (i.e., stream 3 uses seed 3).



When you use a specific seed value (e.g., 17), ProModel produces a unique sequence of numbers to use each time you apply that seed value. This allows you to maintain the consistency of some model elements and permit other elements to vary. (To do this, specify one random number stream for the set of activities you wish to maintain constant, and another random number stream for all other sets of activities.) In fact, because each seed value produces the same sequence of values every time, completely independent model functions must use their own streams. For example, Arrival distributions specified in the Arrival Module should have a random number stream used nowhere else in the model. This will prevent activities that sample random stream values from inadvertently altering the arrival pattern (i.e., the activities will not affect the sample values generated from the arrival distribution).

Please note

The random number generator is a prime modulus multiplicative linear congruential generator. The C code implementation for most of the random variates was written by Stephen Vincent and based on the algorithms described by Law and Kelton (see the "Bibliography" on page 606).

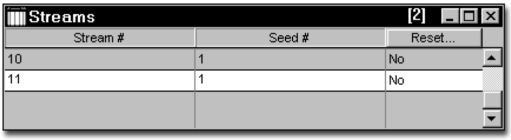
Stream Example

The following example shows one reason why multiple streams are useful.

Two machines, Mach1 and Mach2, are to go down approximately every 4 hours for servicing. To model this, the frequency or time between failures is defined by a normal distribution with a mean value of 240 minutes and a standard deviation of 15 minutes, N(240,15). (For more information on distributions, see "User Defined Distributions" on page 259). The machines will go down for 10 minutes. Because no stream is specified in the normal distribution, ProModel uses stream number one to generate sample values for both machines. So if the next two numbers in stream number one result in sample values of 218.37 and 264.69, Mach1 will receive 218.37 and Mach2 will receive 264.69. Therefore, the two machines will go down at different times, Mach1 after almost four hours and Mach2 after somewhat more than four hours.

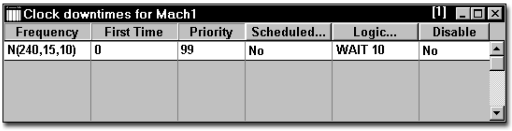
Suppose, however, that the resource to service the machines must service them both at the same time. The machines would have to go down at the same time. Stream number one will not bring them down at the same time because stream number one sends the machine's distributions two different numbers. Using two streams (in the example below numbered ten and eleven) with the same initial seed will ensure that the machines receive the same random number every time. The two streams have the same starting seed value so they will produce exactly the same sequence of random numbers. The first number of stream ten will be exactly the same as the first number of stream eleven; the second numbers will be the same; indeed, every number will be the same.

The Streams window below shows streams ten and eleven with the same seed values.

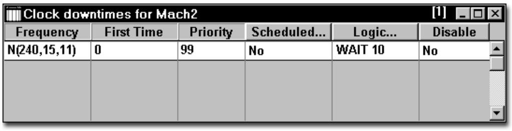


Stream #	Seed #	Reset...
10	1	No
11	1	No

By assigning stream ten to Mach1 and eleven to Mach2, both machines will go down at exactly the same time.



Frequency	First Time	Priority	Scheduled...	Logic...	Disable
N(240,15,10)	0	99	No	WAIT 10	No



Frequency	First Time	Priority	Scheduled...	Logic...	Disable
N(240,15,11)	0	99	No	WAIT 10	No

This first clock downtime for each machine will occur at the beginning of the simulation. After that, the first downtime occurs at 218.37 minutes for both machines. Defining different seed values for the streams would produce different sequences and therefore different downtimes. Using the same stream number for the clock downtimes would also produce different values. But two different streams with the same seed value will bring both machines down at the same times.

Note that if a third machine were to use one of the streams, for example if Mach3 were to use stream eleven, the two machines would no longer go down together. Mach1 would use the first value from stream ten; Mach2 would use the first value from stream eleven; and Mach3 would use the second value from stream eleven. The first time Mach1 and Mach2 went down, they would go down at the same time because the first number in streams ten and eleven is the same. But thereafter they would not. The second time they would go down at different times because Mach1 would receive the second value from stream ten, Mach2 would receive the third value from stream eleven, and Mach3 would receive the fourth value from stream eleven.

Please note

Stream notes:

1. If a stream value is not specified for a distribution, the stream is assumed to be stream one. Stream #1 does not automatically reset after each replication.

2. The stream parameter is always the last parameter specified in a distribution unless a location parameter is also specified. See "Distribution Functions" on page 437 for details.

Material Handling Systems

The following section provides advanced techniques to model Material Handling Systems.

Crane Systems

To implement cranes in ProModel, the Path Network module allows three types of networks: non-passing, passing, and cranes. From this module, you can lay down and orient a crane envelope, define the bridge separation distance, and define the graphics for rails and bridges in the bay. Additionally, the Resources module allows you to define bridge and hoist speeds, and hoist graphics for one or more cranes operating in the same bay.

Creating Multi-Bridge Crane Systems

Modeling multiple cranes operating in the same bay has never been so easy!

User defined bridge separation distances, extensions to regular dynamic resource usage statements, and a set of new priority rules have been introduced in order to let you manage conflicting movements in the same envelope zone.

Crane Envelope The parallelogram-shaped area represented by a crane type path network, bounded by two rails and the lines connecting the endpoints of those rails. The lines connecting the endpoints of rails are effectively the two extreme positions of the center-lines of any bridges operating within the envelope. ProModel uses the end of one of the rails as the envelope origin to serve as a reference point for all logical distance calculations within the envelope.

Bridge Separation The minimum distance you want to maintain between center-lines of two neighboring bridges. This distance is not related to how close a bridge center-line can get to one of the extreme endpoints of the envelope. **Caution:** if bridge B tries to move to a node N that lies very close to the left end of the envelope, and bridge A is to the left of bridge B, a run-time error will occur if the space remaining between node N and bridge A is less than the bridge separation distance. To avoid such problems, define the crane envelope wide enough to allow sufficient space beyond any serviceable nodes.

Crane Priorities, Preemption & Bridge Bump-Away

ProModel associates two types of priorities with the operation of crane resources: (1) *resource request priority*, and (2) *crane move priority.*

Resource Request Priorities Used to resolve any conflicts that arise between multiple tasks requesting the same resource at the same time. This priority scale follows the rules that regular dynamic resource priorities do (i.e., 10 levels of 100). Use this priority to preempt crane resources from lower priority tasks to serve higher priority tasks.

Crane Move Priorities Used for multiple bridges operating in the same envelope, to decide which crane bridge has priority over another while moving. This priority scale is also in the range [0...999], but does not have any preemption levels. Any higher value has overriding priority over a lower value. You can assign priorities to crane resources temporarily, on a task-by-task basis via extra parameters in the GET, JOINTLY GET, USE, and MOVE WITH statements. Therefore, you can only use priorities for cranes that are either moving to respond to a resource request or moving to deliver a part. Moving to downtime, off-shift or break nodes has the default (0) move priority. Moving to park has the lowest move priority (the same as an idle crane).

You can further sub-divide crane move priorities into three categories:

- **Task Move Priority** Assigned by the modeler via parameters passed to resource usage statements (GET, MOVE WITH, etc.). Used as the base move priority for travel to pick up and travel to deposit.

- **Claim Priority** If crane A's bridge is headed towards its ultimate (task) destination, the claim priority is equal to the task move priority of crane A. Otherwise (if crane A's bridge is moving under the influence of crane B), claim priority is equal to the claim priority of the claim inducer bridge (bridge B).
- **Effective Claim Priority** Applies only to envelopes with three or more cranes. If multiple crane bridges are moving in the same direction with overlapping claims, the effective claim priority of any bridge in the overlapping region is the maximum of all the claim priorities of those bridges.

Crane Operations

The operation of cranes in ProModel has been designed to closely resemble the real-life operation of cranes. Under most circumstances, you need not be concerned with how ProModel handles crane movements. The following operational specifications are intended as a reference only and are not essential in order to model cranes using ProModel.

Crane Animation

During a simulation run, entities picked up by a crane appear graphically on the entity spot for 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.

Crane Specifications

When you define cranes for your model, consider the following:

- **Crane Graphics** In the Resources module, the selected graphic represents the hoist. You may define, edit, and size multiple graphics for the same hoist the same way you create multiple graphics for other resources.
- **Units & Multiple Cranes** Crane resources limit you to one unit. When you define multiple cranes operating in the same envelope, define them as separate resources—each assigned to the same crane network.
- **Downtimes** In addition to the clock and usage downtimes you may define for cranes, you can assign cranes to shifts in the same way you assign other resources. When a crane goes down, it moves to the downtime node specified in the DTs dialog. If you do not associate a node with the crane downtime, the crane goes down at its current position.
- **Specifications** Cranes require separate values for bridge and hoist speed and acceleration. You must enter Speed (Empty), Speed (Full), and Accelerate and Decelerate values for bridge movement along the rails and hoist movement across the bridge. In each field, the format is Bridge Value, Hoist Value (e.g., if you enter "150, 20" in the speed field, the bridge moves at 150 and the hoist at 20 feet per minute). Although you must enter speed values, acceleration and deceleration values are optional (if you leave them blank, ProModel assumes the crane to have an infinite acceleration and deceleration capability. (Hint: this provides better run-time computational efficiency).
- **Crane Searches** Define Work and Park searches in the same way you define searches for other resources.
- **Node Logic** Define entry and exit logic for cranes at nodes in the same way you define node logic for other resources.

Handling Zone Claims In Multi-Crane Envelopes

If a crane is in one of the following states, it will be referred to as "immovable at a node." No other crane will be able to push this crane away, regardless of its priority.

- Picking up or depositing an entity.
- Being used at a location.
- Having a downtime, off-shift time, or break time.

Extended Movement Zone The net zone that it needs to traverse ± bridge separation distance (± depending on the direction that it will go).

Before starting to move, any <crane A> attempts to claim its extended movement zone (zone A) for the entire movement duration.

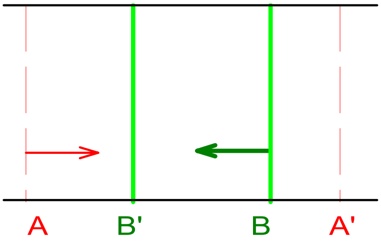
ProModel can reject the claim attempt of crane A for zone A for several reasons:

- If there are immovable cranes in zone A
- If there are any cranes in zone A with move priorities greater than or equal to[1] the move priority of crane A, if they are going in the opposite direction, or they are going in the same direction with conflicting destinations.[2]

Case Examples of Zone Claims

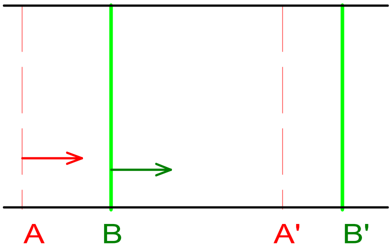
The following cases illustrate specific situations with examples:

Case 1



Bridge B starts moving first, from [B] to [B'] and claims the zone [B'~B] in the ← direction. Bridge A wants to move from [A] to [A'] in the opposite direction. If the move priority of A is less than or equal to the move priority of B, we let bridge A start moving towards [B' - bridge separation] only. If A has a higher move priority, B is stopped immediately and sent back to [A' + bridge separation]. In the latter case, A may queue up behind B.

Case 2



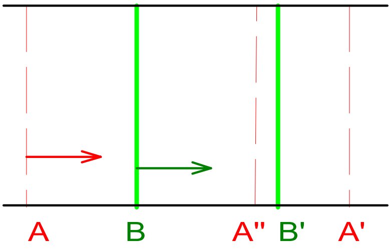
Bridge B starts moving first, from [B] to [B'], then bridge A wants to move from [A] to [A']. Bridge B claims zone [B~B'] in the → direction. Since bridge A wants to move in the same direction and their destinations do not conflict (assuming that B'-A' is greater than the bridge separation), bridge A moves all the way. If B has already cleared the zone [A~(A' + bridge separation)] by the time A wants to start moving, A moves independently. Otherwise, A may queue up behind B. (If the distance B'-A' is less than the given bridge separation and the move priority of A is greater than the move priority of B, bridge A

[1] Special case treatment not ">=" but strictly ">" if all of the following conditions hold: (1) if the bridge that A is trying to oppose is stationary, (2) unless crane A is trying to move just to park (if A has a task or if it is trying to move out of the way of another bridge), (3) if the one A is trying to push is idle or its bridge was blocked before reaching its destination.

[2] Destination Conflict Test: If sign (B' - A' - sign(existing claim)*(bridge_separation)) = sign (existing claim) then not conflicting.

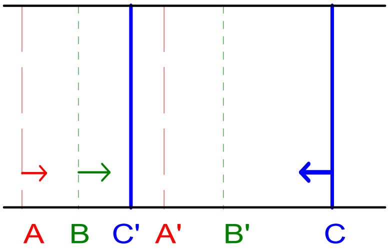
will push away bridge B, and crane B will not respond to the request while passing over [B']).

Case 3



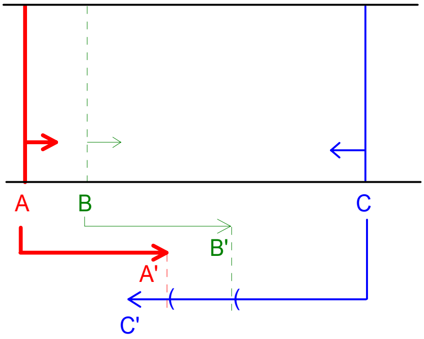
Bridge B starts moving first, from [B] to [B'], then bridge A wants to move from [A] to [A']. Bridge B claims the zone [B~B'] in the → direction. Since bridge A wants to move in the same direction and their destinations do conflict, priorities will be considered. If the move priority of A is not greater[1] than the move priority of B, bridge A starts moving towards [A" = B' - bridge separation] only. Otherwise, A will push B away, and B will not respond to the request while passing over [B']. In both cases, A may queue up behind B.

Case 4



Bridge B starts moving first from [B] to [B'], claiming zone [B~B'] in the → direction. Bridge A wants to move from A to A' and starts moving behind B. Bridge C wants to move in the opposite direction with a higher move priority than both of the other bridges. ProModel interrupts bridges A and B and lays a new course for both bridge B [B" = C' - bridge separation] and bridge A [A" = B" - bridge separation]. In this case, B may queue up behind A, and C behind B.

Case 5



This is an extension of Case 4. Assume that the move priorities for crane bridges A, B, and C are 3, 1, and 2 respectively. When bridge C tries to claim zone [C'~C] in the ← direction, it faces an “effective priority” of 3 and is unsuccessful in its original claim, so it starts moving towards [B' + bridge separation]. Bridge B goes outside the effect of bridge A in the second part of its movement, [(A' + bridge separation) ~ B'] at time t0. However, bridge C does not override bridge B and start moving towards [A' + 2 * bridge separation] until the first “reclaim trigger event[2]” after time t0.

If ProModel rejects the original claim of crane A, crane A claims the largest available portion of the original zone, and implements the procedure described below to move its bridge through the subset zone. The crane waits until a “reclaim trigger event” occurs, and then repeats the claim attempt to let its bridge traverse the remaining

[1] In this special case, bridge A’s claim priority competes with the task move priority of bridge B.

[2] Discussed later in this section.

distance. The hoist, on the other hand, moves directly to its relative destination on the bridge, independent of the bridge zone claim results. In case of bridge interruption, the hoist does not stop moving. Therefore, for any particular crane move, the combined trajectory of hoist and bridge may change depending on zone availability at the time of the claim.

If crane A accepts the claim attempt, all cranes inside the claimed extended zone bump away at the time of the claim. This claim also prevents any other cranes with move priority less than or equal to the move priority of A from entering the claimed zone in conflicting directions.

Zone Claim Notes

Zone Claim Notes

1. A crane cannot do a pickup or deposit while another crane pushes it away, even though the request might be on its way.
2. When you interrupt a crane bridge and ask it to make a new claim (or update its claim), its speed is reset to zero and ProModel re-evaluates its mobility characteristics. ProModel recalculates the move time with these new parameters. This rule also applies to bridges that are moving when a bridge behind them with a higher priority and a conflicting destination makes a new claim to go in the same direction. See Case 3 (when priority of A is greater than priority of B) for an example.

Treatment of Potential Queue-Up Situations

For all bridge claims with the potential of having two bridges queue up after one another (mentioned in the example cases above), the following rule applies:

The acceleration rate (***a***), maximum speed (***s***) and deceleration rate (***d***) of the following bridge (***F***) are compared with the ***a***, ***s***, and ***d*** of the leading bridge (***L***). If all of the following conditions are true, bridge ***F*** moves independent of bridge ***L***.

Conditions for No Queue-Up

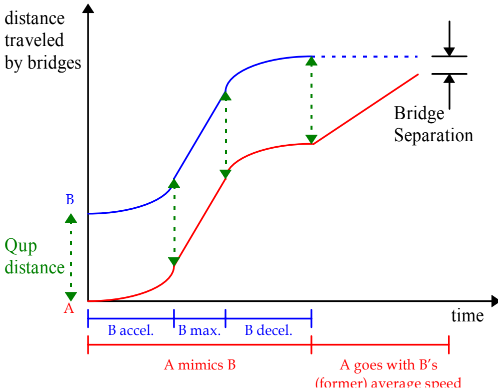
1. $a_F <= a_L$ The following bridge cannot accelerate faster than the leading bridge.
2. $s_F <= s_L$ The following bridge's maximum speed cannot be greater than the leading bridge.
3. $d_F >= d_L$ The following bridge can decelerate equally or faster than the leading bridge.

If any one of the above conditions fail, bridge F moves at the same speed as bridge L, maintaining the distance that existed between them at the time F started moving.

Please note

Condition 3 usually creates a deviation from the typical real life situation, but it is necessary to avoid time consuming computations that can adversely affect the run-time performance of models with crane resources. To avoid unrealistic queue-ups when modeling multiple bridges with different mobility characteristics in the same crane envelope, make either no bridge deceleration entry (infinite deceleration), or use the same overall average deceleration rate for all crane bridges in that envelope.

Bridge Motion after Queue-Up



If bridge A is queued up behind bridge B, bridge A goes through up to two stages of movement. The first stage occurs throughout the movement of bridge B. Bridge A mimics bridge B's motion, maintaining its original distance from bridge B.

The second stage occurs after bridge B stops. Bridge A continues traveling with the former average speed of bridge B until it reaches its destination.

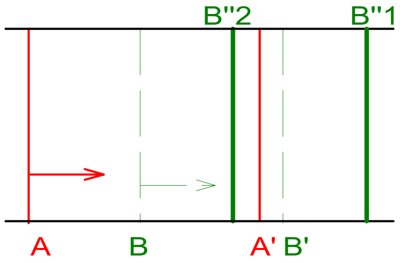
If bridge B is moving to accomplish its own task (as opposed to just moving out of A's zone), then bridge A might stop earlier, without going through the second stage. In this case, the total traveling time for bridge A is calculated as if bridge A were traveling with bridge B's average speed.

When a bridge arrives at its destination, it triggers a chain queue-up re-evaluation that penetrates the crane envelope in the opposite direction that the stopped bridge was moving. This chain re-evaluation handles cases in which there were multiple bridges queued up behind the stopped bridge.

Moving Idle Cranes can Respond to Resource Requests

A crane can be idle and moving at the same time (moving to park or being pushed away by another). Independent of its current position and movement direction, the crane still has the capability to respond to resource requests coming from anywhere in the envelope. If allocated to a task with high enough travel-to-pickup move priority, the crane may reverse direction or terminate its move early, bumping other crane bridges on its way.

Whenever a crane bridge changes its claim, it checks to see if there are any bridges behind it that are going in the same direction with overlapping claims. If there are any, the newly claiming crane triggers a chain queue-up re-evaluation towards the following cranes. The following example demonstrates this case.



At time t0, crane A is called for a task and starts moving to [A'] with move priority mpA, pushing idle bridge B to [B' = A' + bridge separation]. At time t1 (bridge A and B are still moving), a request for crane B comes from [B" (1 or 2)] with move priority mpB. Bridge B is allocated to the task immediately since it is available at the time.

If the new destination of bridge B is [B"1], the new claim of bridge B does not conflict with the existing claim of bridge A, so the move priorities are not significant. ProModel re-evaluates all motion values for crane B (acceleration, deceleration, empty and full speeds for both the hoist and the bridge) and crane B starts moving to its new destination. If the bridge motion expressions for crane B have changed their values between t0 and t1, the queue-up status between bridges B and A (and any others behind A) may change, so

ProModel performs a chain queue-up re-evaluation.

If the new destination of bridge B is [B"2], the new claim of bridge B conflicts with the existing claim of bridge A, so ProModel considers the move priorities:

- If mpB <= mpA, then B does not have enough priority to stop A before A gets to its destination. Bridge B will have to go on moving until [B'] and wait until it can claim the region [B"2 ~ B'] in the → direction.
- If mpB > mpA, then B does have overriding priority over A. Bridge B stops bridge A at [B"2 - bridge separation], starting a chain move termination which tells bridge A to stop any other followers with the appropriate bridge separation distances between each. In this case, bridge A (and other possible followers) will have to wait until the next reclaim trigger event to attempt reaching their original destinations.

Reclaim Trigger Events

Unsuccessful claims of all crane bridges in an envelope are re-attempted when a reclaim trigger event occurs in that envelope. The following events are considered to be triggers for claim re-evaluations in the related envelope:

- The duration of a USE <crane> statement expires.
- You free a crane using the FREE statement.
- A crane completes a deposit (unless you specify the KEEP option).
- A crane completes a pickup and the crane is about to start moving to deliver.
- A crane downtime ends.
- You preempt a crane resource.
- A bridge arrives at its task destination (before elapsing pickup/deposit time).
- There are no moving task bridges left (for envelopes with 3 or more cranes only).

In the first six cases, a crane changes state from “immovable” to a finite move priority. The seventh case may change the “effective move priorities” of other cranes. ProModel requires the eighth case for envelopes with three or more bridges with overlapping operation zones: It ensures that two or more unsatisfied, opposing claims with the same move priority will resolve their conflict based on earlier task assignment time instead of locking up the crane system. Hence, all of the above cases create a potential to realize previously unsuccessful claims.

The reclaim trigger events for each envelope echo in the trace together with the triggering reason for model debugging purposes. The results of reclaim attempts for each crane resource and its effects on other crane resources in the same envelope are also displayed.

Other Crane Operation Rules

1. If an entity currently possesses a crane, all move logic for the entity must use a MOVE WITH statement (until you free the crane)—the entity cannot move without the crane.
2. Although an entity cannot possess more than one crane at a time, the entity may possess a crane and a resource of another type.
3. If a crane is in one of the following states, another crane cannot push it away.
   a) Picking up or depositing an entity.
   b) Being used at a location.
   c) Experiencing a downtime, off-shift time, or break time.
4. The hoist moves to its relative destination on the bridge independent of the bridge zone claim attempt outcomes. The hoist does not stop for bridge interruption so, for any particular crane move, the combined trajectory of hoist and bridge may change depending on zone availability at the time of the claim.
5. When a crane pushes away an idle crane, the idle crane comes under the control of the

work and park search lists defined for the node closest to its bridge.

Crane Operations Notes

ProModel allows definition of resource points for crane resources on envelope nodes to achieve the three dimensional look for “lowering the hoist” upon arriving at a node. Since cranes can only have one unit, only the first resource point for any crane at any node is meaningful. Also, ProModel only uses the resource points if the crane resource arrives at the node to perform a task. In other words, the hoist does not lower if the crane arrives at a node to park or just ends up there after a bump.

In general, once ProModel makes a match between a resource request and a resource of the requested type, the resource is allocated for that job and the simulation engine does not go on looking for other matches that can actually reach to the requester sooner. Since cranes blocking other crane resources is a common phenomenon in multi-bridge envelopes, extra care should be used while designing the active operation zones and resource request logic in such models.

Nodes, Work, and Park Searches

ProModel “registers” all cranes to their home nodes at the beginning of a simulation run. When a crane starts to move, ProModel erases registration, and re-establishes it when the crane stops, according to the following rules:

1. Cranes assigned to a task (pick up, drop off, down) remain unregistered throughout intermediate stops and register to the task destination node upon arrival.
2. Idle cranes that are bumped away, or cranes that were unsuccessful in claiming the entire bridge region to go parking register themselves to the node closest to their bridge when they stop. If there are multiple nodes closest to the bridge, ties are broken by the minimum distance to the hoist and then the order of appearance in the nodes edit table. If the hoist is still moving at the time the bridge stops, the registry procedure executes by considering the destination of the hoist.

Please note

Only users can instruct cranes to park on nodes. However, as a result of bump-aways, a crane can end up standing still at positions that do not necessarily correspond to nodes. In any case, ProModel registers an idle stationary crane to a node and places it under the control of work and park searches of that node.

Cranes follow the same rules as regular dynamic resources while searching for assignments (returning to preempted jobs, exclusive & non-exclusive work searches, requests coming from nodes not included in the work search, etc.). There are some additional steps to handle interactions with other cranes operating in the same envelope. Upon completion of its task, a crane goes through these extra steps:

1. Lets other cranes in the same envelope re-attempt their claims, possibly generating an induced claim on its bridge.
2. Even though other cranes might bump away its bridge, the crane executes the work search defined for it at its current node. (If requests are waiting at different locations of the same work search record, ProModel takes the resource request priorities into consideration.)
3. If it was not allocated to a task or bumped away, the crane executes the park search defined for it at its current node, or attempts going home if return home is checked.

Please note

A crane bridge moving to park cannot bump other idle bridges out of its way. In such a case, the bridge goes as far as it can towards its destination and registers itself to the closest node upon stopping (using the rules discussed above).

Crane Resource Statistics

The statistics reported for cranes are very similar to regular dynamic resource statistics. The following definitions highlight the differences in the interpretations.

In Use Includes the following states:

- Stationary use at a location via GET, JOINTLY GET, and USE statements.
- Gross move time while delivering — actually carrying an entity from one location to another.
- Pickup and deposit times before and after the move to deliver.

Travel To Use Is the gross move time to start being used — traveling to respond to a request for either stationary use or use to pickup and deliver an entity to another location.

Gross Move Time May include the following times:

- Moving towards the actual destination.
- Possibly moving away from the destination, when bumped away by a higher priority bridge in the same envelope.
- Stationary blockage times during which the bridge is waiting for a higher move priority bridge in the same envelope to lower its move priority.

Blocked Time Tally blocked time by subtracting the internally computed, hypothetical net traveling time from the actual incurred gross move time. ProModel defines the net time as the time a crane would have taken to get to its destination if it were the only bridge in its envelope.

Please note

The first component of gross travel time (moving towards the actual destination) may be greater than the net travel time, since it may include going back towards the destination after another crane bumps it away. It may also involve deviations from the bridge's original motion values when queued-up behind slower bridges.

Special Case—Blocked Time Accrual for Moving to Park

Remember that an idle crane registers itself to the closest node whenever its bridge stops. If the hoist is still moving, ProModel uses the destination of the hoist to break any ties between nodes that are equidistant to the bridge position.

In case of an unsuccessful claim attempt to move to park, the crane bridge needs to stop before reaching the instructed park node, and ProModel ignores the hoist travel for statistics purposes. The “travel to park” statistic terminates at the time the bridge stops, not when bridge or hoist stop time reaches its maximum value. In this case, the net moving time to park computes as the bridge move time from current position to the endpoint of the claim, using original bridge motion values for the crane resource. Blocked time can accrue if this bridge queues up after a slower bridge that is already moving (remember that a crane moving to park cannot bump another crane).

Conveyors

A conveyor is any moving track, belt, chain, or roller which transports parts from one location to another. This section will help you better understand conveyors and how to model them using ProModel.

ProModel determines the capacity of a conveyor by its speed and load spacing rather than a stated capacity. Specifically, capacity is a function of the minimum allowable interload spacing on a conveyor (which is the length of a queue position in the case of accumulation conveyors) and its length. It may be desirable, however, to impose a limit to the number of loads permitted on the conveyor at any one time.

Conveyors usually do not pick up and drop off loads as in the case of lift trucks, AGVS, and cranes. You must place loads onto and remove them from conveyors.

Depending on the nature of the conveyor and its operation, modeling a conveyor can be straightforward or complex. For single conveyor sections, modeling is very simple. Conveyor networks, on the other hand, give rise to several complexities (recirculating, merging, etc.) that become more difficult to model.

Conveyor Simulation Benefits

Simulating your conveyor system will help you address design and operational issues, answering the following questions:

- What is the minimum conveyor speed that still meets rate requirements?
- What is the load rate capacity of the conveyor?
- What is the load delivery time for different activity levels?
- How much queuing do accumulation conveyors need?
- How many carriers do trolley or power-and-free conveyors need?
- What is the optimum number of pallets to maximize productivity in a recirculating environment?

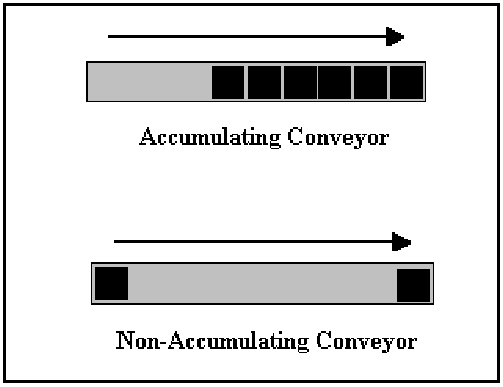
Conveyor Types

You may represent conveyors as one continuous span of randomly spaced parts (e.g., belt conveyor) or as intermittent spaces or carriers located at fixed intervals (i.e., trolley conveyor).

Accumulating & Non-Accumulating

In addition to continuous or intermittent conveyors, some conveyors permit loads to accumulate or queue if conditions impede forward motion progress. ProModel refers to these conveyors as accumulating conveyors. Other conveyors provide no queuing and halt all activity if forward progress of the leading part or load stops. ProModel refers to these types of conveyors as non-accumulating conveyors.

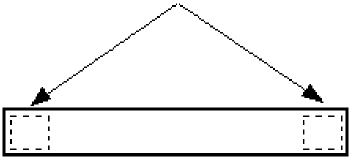
The following figure illustrates how entities can continue to accumulate after the leading load stops on an accumulating conveyor. How a non-accumulating conveyor can accept one additional entity if the leading entity stops, however it is unable to advance any further since the entire conveyor stopped.



The different combination of continuous or intermittent conveyors and transport or accumulation conveyors result in the following four different kinds of conveyors that ProModel is capable of modeling.

Type	Example
Continuous, Non-Accumulating	Belt, chain
Intermittent, Non-Accumulating	Tray, trolley
Continuous, Accumulating	Roller
Intermittent, Accumulating	Power-and-free, towline

For best results, you should model carriers or part holding devices on intermittent, accumulating conveyors as though the devices were entities waiting for you to load and unload other entities to and from them (see "Load" is ProModel User Guide) as you move parts. An alternative but similar method is to use a JOIN statement and then specify two routings when you want the part removed (see Example E in "Modeling Conveyor Systems" in the next section). Yet another way of modeling parts with carriers on a conveyor, is to simply set a part attribute equal to a particular value if the part is on a carrier. If the part is not on a carrier, assign a different value to the same attribute. Operations may occur at either the beginning or end of a conveyor as shown here.



On-conveyor operations must occur at the beginning or end of a conveyor section.

Single Section Conveyors

Conveyors used for simple queuing or buffering often consist of a single stretch or section of conveyor. Loads enter at one end and exit at the other end. Load spacing is generally random. These types of conveyors are generally quite easy to model. The modeler merely defines the length and speed and states whether the conveyor is an accumulating or non-accumulating conveyor.

Conveyor Networks

A conveyor network consists of two or more sections that are connected together to enable part merging, diverting, and recirculating. In such instances, a conveyor may have one or more entry points and one or more exit points. Furthermore, part entry and exit may not always happen at the beginning or end of the conveyor. In ProModel, loads or entities must always enter at the beginning and exit at the end of conveyor networks. To model merging or branching in the middle of the conveyor, break the conveyor into smaller sections.

Modeling Conveyor Systems

A conveyor system is a combination of one or more types of conveyors which interact to move parts. Such a system may include branching, merging, and looping. The process of dividing conveyors into logical sections is fairly straightforward if you understand what constitutes a conveyor section.

For modeling purposes, a conveyor section is defined as any independently driven segment of conveyor of the same type. Conveyor turns are irrelevant as long as the conveyor remains unchanged and both are actuated or stopped together.

In ProModel, an accumulating conveyor section must end at the point where parts are permitted to accumulate or back up. This generally occurs at a transfer point or process location.

Modeling Conveyors

While not essential, it is visually more effective to draw conveyors to scale on the layout. You should also size entities traveling on a conveyor to scale when you create them. Setting the length of a conveyor different from the scale length or conveying entities that have a length or width that is not to scale will result in the appearance of load overlapping or excessive load spacing. The underlying logic and simulation is still valid, regardless of whether the graphics are to scale.

Loads may appear to jump when transferring from the end of one conveyor to another conveyor. This is a result of ProModel not allowing a load to transfer onto a conveyor until there is room for the load. Including a transfer time in the Move Logic column of the Routing edit table can result in smooth movement between conveyor sections. Defining a variable called Transfer_Time and using that variable rather than entering the actual time allows for experimentation with various conveyor speeds with minimal modeling changes.

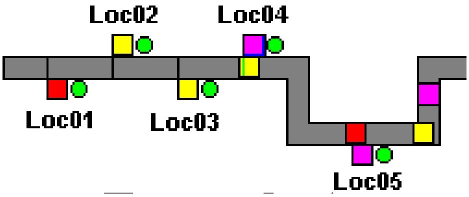
Clock precision is also important. Note that movement of one foot at 40 feet per minute will take .025 minutes. If clock precision is set at .1 minutes, truncation will result in zero time. Make sure you check your model for the correct clock precision. For most conveyors, a clock precision of .001 is adequate. ProModel may require greater precision for high-speed conveyors where speeds are greater than 200 fpm.

Modeling conveyors in ProModel is quite easy. Select the conveyor icon in the Locations Graphics window and place the beginning and ending points of the conveyor using the mouse. Clicking the right mouse button will end the conveyor. Left-click to add or delete a joint and allow the conveyor to bend.

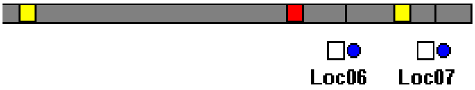
Some users prefer to display the grid and select the Snap to Grid option in the View menu. Others will leave Snap to Grid off. You may find that the precise positioning of a conveyor is easier when you turn Snap to Grid off and set the Zoom sufficiently high. When you need to change the type of conveyor to accumulating, double-click on the conveyor icon in the layout, select the Conveyor Options button, and select Accumulating from the Conveyor Options dialog.

The following six examples show various ways of modeling conveyors:

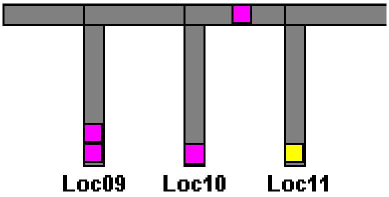
Example A Routes one or more parts from one or more non-conveyor locations to any conveyor location.



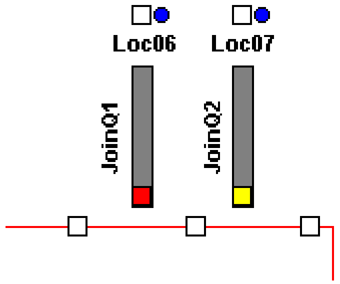
Example B Routes one or more parts from any conveyor location to one or more non-conveyor locations.



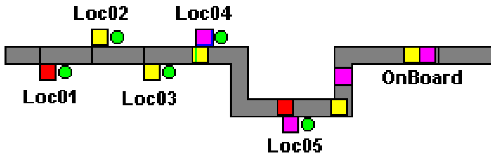
Example C Routes single parts from any conveyor location to one or more other conveyor locations.



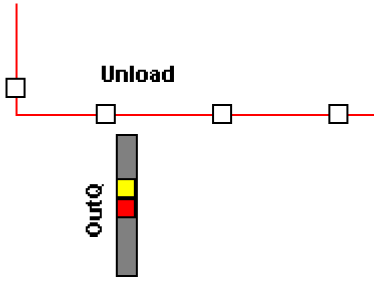
Example D Joins one or more parts from one or more non-conveyor locations to another transporting entity (e.g., a pallet or carrier) at any conveyor location.



Example E Accumulates multiple parts coming from one or more non-conveyor locations onto a conveyor location.



Example F Splits parts at a conveyor location sending one or more parts to a non-conveyor location and a single part (e.g., a pallet or carrier) onto the conveyor.



Automated Guided Vehicle Systems

An automated guided vehicle system (AGVS) is a path network along which computer-controlled, driverless vehicles transport loads. One of the modeling requirements of AGVS is to accurately describe the method for controlling traffic. You can usually accomplish this in one of two ways:

- Zone blocking
- On-board vehicle sensing

Zone Blocking The most common method of traffic control, it involves placing control points along the guide path. Each point usually allows only one vehicle to access it any one time, thus blocking any other vehicle from entering any segment of the path connected to that point. Once the vehicle leaves a control point to travel to the next point on the path, any vehicle waiting for access to the freed control point can resume travel.

On-Board Vehicle Sensing Works by having a sensor on-board the vehicle that detects the presence of a vehicle ahead of it and stops until it detects that the vehicle ahead of it has moved.

Modeling AGVS

Modeling an AGVS is very similar to modeling an industrial vehicle, such as a lift truck, (which it is in a sense) except you control the operation more and there exists less freedom of movement. Paths are generally unidirectional and do not allow vehicle passing.

One of the challenges in modeling an AGVS is finding the shortest routes between any two stops in a system. ProModel provides built-in capability to automatically determine the shortest routes between points in a complex network.

AGVS Simulation Benefits

Simulating your AGV system will help you address design and operational issues, answering the following questions:

- What is the best path layout to minimize travel time?
- Where are the potential bottleneck areas?
- How many vehicles do you need to meet activity requirements?
- What are the best scheduled maintenance/ recharging strategies?
- Which task assignment rules maximize vehicle utilization?
- What is the best idle vehicle deployment to minimize response time?
- Is there any possibility of deadlocks?
- Is there any possibility of collisions?

Manual Material Handling Systems

Manual material handling systems (MMHS) are probably the most common systems for handling, moving, storing, retrieving, and managing materials. ProModel easily allows you to model manual labor. With shifts, breaks, and downtimes (illness, training, etc.), model people as resources.

Whether carrying parts from one station to another or stocking shelves in a grocery store, people can be modeled as resources to do just about anything in ProModel. Such resources are dynamic and move along assigned path networks.

Since people can think, you can model them very much like automated guided vehicles with onboard vehicle sensing.

MMHS Simulation Benefits

Simulating your manual labor material handling system will help you address design and operational issues, answering the following questions:

- What is the best path layout to minimize travel time?
- Where are the potential bottleneck areas?
- How many people do you need to meet activity requirements?
- What are the best shift and break schedule strategies?
- Which task assignment rules maximize utilization?
- What is the best idle person deployment that minimizes response time?

Industrial Vehicles

Industrial vehicles include all push or powered carts and vehicles that generally have free movement. Utilize powered vehicles such as lift trucks for medium distance movement of batched parts in a container or pallet. For short moves, use manual or semi-powered carts. Single load transporters are capable of moving only one load at a time from one location to another. Such devices are fairly straightforward to model because they involve only a single resource and a single destination for each move.

Multiple load transporters, on the other hand, can move more than one load at a time from one or more sources to one or more destinations. Defining capacity and operation for multiple load transporters can be extremely difficult since there may be special rules defining when to retrieve additional loads and when to deliver the loads already on board.

Modeling Industrial Vehicles

Modeling an industrial vehicle involves modeling a resource that moves along a path network. Paths are typically open aisles in which bi-directional movement is possible and passing is permitted. You may also need to define deployment strategies (work searches, idle vehicle parking, etc.).

Since people generally operate industrial vehicles, they take breaks and may be available only during certain shifts. So in addition to modeling the vehicle, you may need to model an operator that takes breaks and works on shifts or you may just want to assign shifts and breaks to the vehicle resource itself, in effect, modeling the vehicle and the operator as one resource.

Industrial Vehicle Simulation Benefits

Simulating your industrial vehicles will help you address design and operational issues, answering the following questions:

- What is the required number of vehicles to handle the required activity?
- What is the best deployment of vehicles to maximize utilization?
- What is the best deployment of vehicles to minimize response time?

Automated Storage/Retrieval Systems

An automated storage/retrieval system (AS/RS) is “a combination of equipment and controls which handles, stores, and retrieves materials with precision, accuracy, and speed under a defined degree of automation” (The Material Handling Institute 1977). The goal of an AS/RS is to provide random, high density storage with quick load access, all under computer control.

Modeling AS/RSs

At a simple level, an AS/RS move time may be modeled by taking a time from a probability distribution that approximates the time to store or retrieve a load. More precise modeling incorporates the actual crane (horizontal) and lift (vertical) speeds. Each movement usually has a different speed and distance to travel which means that movement along one axis is complete before movement along the other axis begins. From a modeling standpoint, it is usually only necessary to calculate and model the longest move time.

In modeling AS/RS, the storage capacity is usually not a consideration and the actual inventory of the system is not modeled. It would require lots of overhead to model the complete inventory in a rack with 60,000 pallet locations. Since it is primarily only the activity that is of interest in the simulation, actual inventory is ignored. In fact, it is usually not even necessary to model specific stock keeping units (SKUs) being stored or retrieved, but only distinguish between load type insofar as it affects routing and subsequent operations.

Modeling an AS/RS as a Vertical Bridge Crane

One way to model an AS/RS with greater accuracy is to use the bridge crane construct. An AS/RS is really just a vertical crane with the same characteristics as a bridge crane.

AS/RS Simulation Benefits

Simulating your AS/RS will help you address design and operational issues, answering the following questions:

- How many aisles do you need to handle the required activity?
- What is the best sequence of stores and retrievals to maximize throughput?
- What is the best stationing of empty S/R machines to minimize response time?

Modeling Tips

The following section provides advanced Modeling Tips.

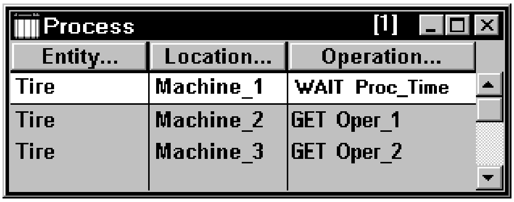
Using Entity Attributes

Entity attributes are a powerful construct that enable a user to “tag” entities with characteristics to distinguish them from other entities of the same name.

Entity attributes are place holders associated with individual entities which usually contain information about that entity. Attributes may contain integers or real numbers. Unlike variables, attributes are not global. To define attributes, use the Attributes module found under the More Elements option of the Build menu.

One use of an entity attribute is to represent a processing time. This is especially useful when you base processing times at a location on some characteristic other than entity type, such as size or condition.

Suppose several types of tires enter a machining center that places the steel belts on the tire. The processing time of the tire increases as the diameter of the tire increases. The following example shows that for a tire entering a machining location, the processing time uses the value of the attribute “Proc_time.” You can make the attribute assignment in the “Logic” field of the Arrivals module.

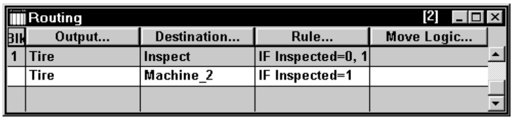


Entity...	Location...	Operation...
Tire	Machine_1	WAIT Proc_Time
Tire	Machine_2	GET Oper_1
Tire	Machine_3	GET Oper_2

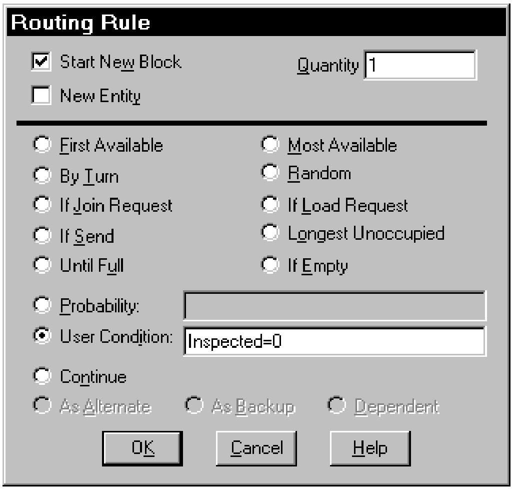
In this example, ProModel attaches the attribute “Proc_time” (short for process time) to the entity, tire, which contains a value representing the process time at Machine_1. You can assign a value to “Proc_time” in the Operation field or, as mentioned above, in the Logic field in the Arrivals module. The latter allows you to assign a value to the attribute for each entity as they arrive in the system.

When using attributes instead of operation times, ProModel substitutes the value assigned to the attribute directly into the Operation field. So, if “Proc_time=4.5” were your attribute assignment and the default units for time were in minutes, the resulting operation time would be 4.5 minutes.

You may also use attributes to determine the routing of an entity. Note that the Routing Rule dialog offers a “User Condition” routing rule. In this field, you can enter an expression that evaluates the value of an attribute for the entity currently at that location. Hence, ProModel selects the routing destination based on the attribute.



Blk	Output...	Destination...	Rule...	Move Logic...
1	Tire	Inspect	IF Inspected=0, 1	
	Tire	Machine_2	IF Inspected=1	



☑ Start New Block   Quantity 1
□ New Entity

○ First Available   ○ Most Available
○ By Turn   ○ Random
○ If Join Request   ○ If Load Request
○ If Send   ○ Longest Unoccupied
○ Until Full   ○ If Empty

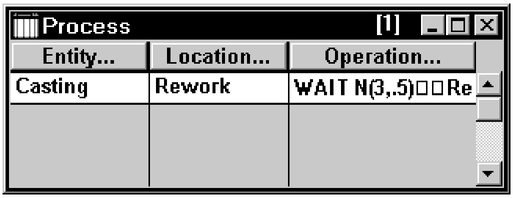
○ Probability:
● User Condition: Inspected=0
○ Continue
○ As Alternate   ○ As Backup   ○ Dependent

In this example, an entity attribute called “Inspected” has a value of zero or one. For uninspected tires, Inspected=0. For inspected tires, Inspected=1. Set the attribute value equal to 1 in the Operation field of the Inspection location to indicate an inspected tire.

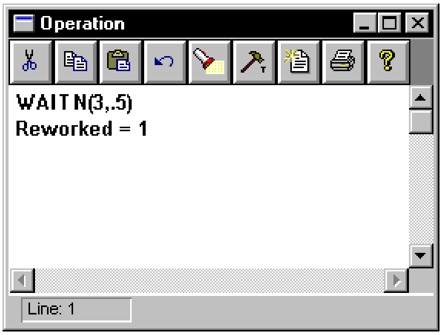
At some other location in the model, ProModel makes a routing decision based the tire's inspection status. The example above shows that the entity, tire, will route to either Inspect or Machine_2 based on the attribute value.

Another use for entity attributes is for assigning entity characteristics such as number of times reworked, color, model type, lot size, order number, etc. You may do this to reduce the number of system entities defined.

In the following example, to show that you reworked a casting, each entity that goes through the location Rework has its "Reworked" attribute set equal to 1.



Entity...	Location...	Operation...
Casting	Rework	WAIT N(3,.5)□□Re



```
WAIT N(3,.5)
Reworked = 1
```

You may assign values to entity attributes in the Operation field in the Processing edit table, the Logic field in the Arrivals edit table, and the Move Logic in the Routing edit table.

Customizing Graphics

Background graphics can enhance the look of your models and provide valuable visual information to convey meaning and lend credibility to your models. Several file types can be imported from various sources including CAD drawings.

How to Import a background graphic:

1. Choose **Background Graphics** from the **Build** menu.
2. Select **Front of Grid** or **Behind Grid**.
3. Choose **Import Graphic** from the **Edit** menu.

Front of Grid When the grid is on, the grid lines will not cover or obstruct the imported graphic.

Behind Grid The imported graphic will be behind the grid lines when the grid is on.

You may paste any *.BMP or *.WMF image copied to the Windows clipboard on the Layout by choosing Paste BMP or Paste WMF from the Edit menu.

Once you select Import Graphic from the Edit menu and select the desired file and type, ProModel imports the image into the layout. Once this procedure is complete, you may select or activate the graphic by clicking with the left mouse button on the graphic. You may then drag the mouse to move the graphic to the desired position on the layout. You may stretch or compress the graphic to the size you desire using any of its four sizing handles.

ProModel allows you to import *.BMP, *.WMF, *.GIF, and *.PCX file types. BMPs and WMFs can also be copied to the clipboard from other applications and pasted directly into the background. To import a CAD drawing, you must save it to one of the file formats described above.

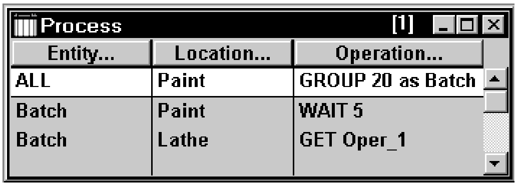
Batching & Unbatching Entities

Batching refers to congregating or consolidating multiple entities of the same or different type for processing or movement purposes. Batching may be either temporary or permanent. For temporary batching, use the GROUP statement. For permanent batching, use the COMBINE statement.

Temporary Batching Using GROUP/UNGROUP

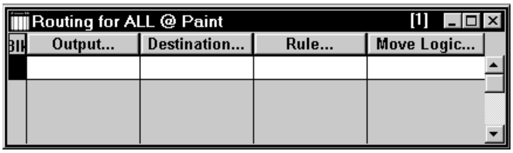
The GROUP statement allows you to group entities together and ungroup them at a later time. You may group entities by individual entity type by defining a process record for the type to group, or group them irrespective of entity type by defining an ALL process record. To combine multiple entity types in which you must control the quantity of each type requires controlling the routing which sends parts to the grouping location. ProModel maintains all of the identities and attributes of the grouped entities and allows them to remain with the individual entities after an UNGROUP command.

To illustrate how GROUP/UNGROUP works, suppose we want to consolidate 20 incoming entities into a group called Batch. If the current location performs no additional operation steps once you group the entities, you can simply use the statement “GROUP 20” to group the 20 entities and then specify an output of 1 Batch in the routing. If, on the other hand, the current location performs additional operation steps after the grouping occurs, use the statement “Group 20 as Batch” and define no output routing for the grouped entities. Instead, the entities become a new entity called “Batch” that needs a process record defined afterwards for the same location where you batched them.



Process [1]

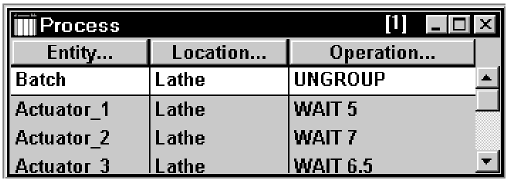
Entity...	Location...	Operation...
ALL	Paint	GROUP 20 as Batch
Batch	Paint	WAIT 5
Batch	Lathe	GET Oper_1



Routing for ALL @ Paint [1]

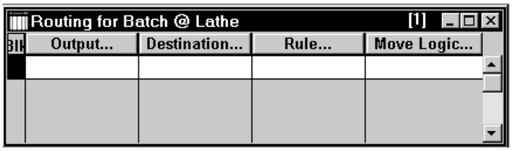
Bl	Output...	Destination...	Rule...	Move Logic...

Once you accomplish the purpose for which you grouped the entities, such as moving them, you can unbatch the entities using the UNGROUP statement as shown next. Note that after the ungrouping takes place, you must define processes for each potential entity that could have been in the group (alternatively you could define an ALL process record).



Process [1]

Entity...	Location...	Operation...
Batch	Lathe	UNGROUP
Actuator_1	Lathe	WAIT 5
Actuator_2	Lathe	WAIT 7
Actuator_3	Lathe	WAIT 6.5

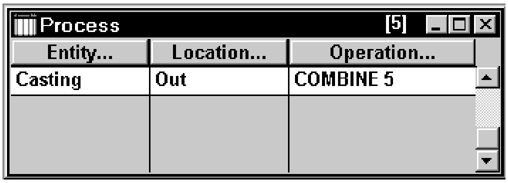


Routing for Batch @ Lathe [1]

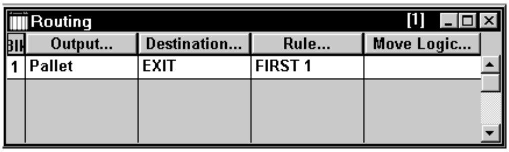
Bl	Output...	Destination...	Rule...	Move Logic...

COMBINE Statement

ProModel uses the COMBINE statement to accumulate and consolidate a specified quantity of entities into a single entity, optionally with a different name. The entities may be the same type of entity or they may be different. Combined entities lose their identities and attributes and you cannot ungroup them later. When defining the location, the location capacity where you use the COMBINE statement should be at least as large as the combined quantity. The following shows the correct syntax for combining 5 Castings into a single entity called Pallet (the location, Out, should have a capacity of at least 5).



Entity...	Location...	Operation...
Casting	Out	COMBINE 5



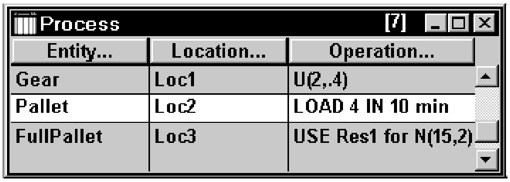
Bl	Output...	Destination...	Rule...	Move Logic...
1	Pallet	EXIT	FIRST 1	

LOAD/UNLOAD Statements

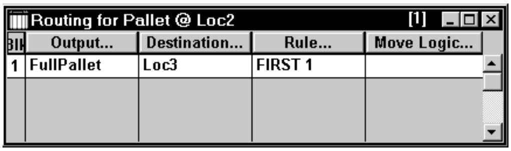
Another method of batching and unbatching involves the LOAD and UNLOAD statements. As with the GROUP statement, ProModel maintains the identities and attributes of the batched entities. Therefore, when you use the UNLOAD statement, each entity still has its attribute assignments.

The difference between LOAD and GROUP is that LOAD requires a “base” entity on which you load other entities that might represent a pallet or container. Also, the entities you are going to load, travel to the loading location on an “If load request” routing rule.

In the following example, four Gears are loaded onto a pallet. Pallet is the entity which issues the LOAD statement. In addition, the example includes a time element (LOAD 4 in 10 min). This means “load 4 gears or wait 10 minutes, whichever occurs first.” If after 10 minutes only 3 gears arrive, the entity loaded with the 3 gears continues.

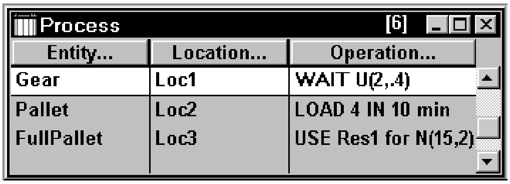


Entity...	Location...	Operation...
Gear	Loc1	U(2,.4)
Pallet	Loc2	LOAD 4 IN 10 min
FullPallet	Loc3	USE Res1 for N(15,2)

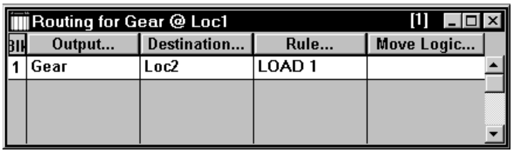


Bl	Output...	Destination...	Rule...	Move Logic...
1	FullPallet	Loc3	FIRST 1	

Note that the loaded entity, Gear, has “LOAD” in the Routing Rule field as shown here. This tells ProModel that the gear only goes to Loc2 when a Load command requests it.



Entity...	Location...	Operation...
Gear	Loc1	WAIT U(2,.4)
Pallet	Loc2	LOAD 4 IN 10 min
FullPallet	Loc3	USE Res1 for N(15,2)



Bl	Output...	Destination...	Rule...	Move Logic...
1	Gear	Loc2	LOAD 1	

Since the example does not specify an entity type as part of the LOAD statement, ProModel will load entity type waiting for the LOAD request. To control which entity types the model loads, use the “IFF” option with the LOAD statement (e.g., LOAD 4 IFF Entity()=gear).

Once the batch moves to its destination, the model uses the UNLOAD statement to break it into individual components (Gear and Pallet). You must define a routing for the entity Gear after the UNLOAD takes place. Typical uses for LOAD/UNLOAD are palletizing operations, AGV systems, and AS/RS systems.

ACCUM Statement

The ACCUM statement is short for “accumulate.” Its use is slightly different than the GROUP and LOAD statements. ACCUM does not “batch together” entities, but rather holds entities at a location until a certain number accumulate. It behaves much like a gate restricting entities until a certain number arrive at the location. After the required number accumulate, ProModel releases them for downstream processing. This allows you to model a certain type of batching which involves simply the accumulation of parts and their subsequent release—all at a single location.

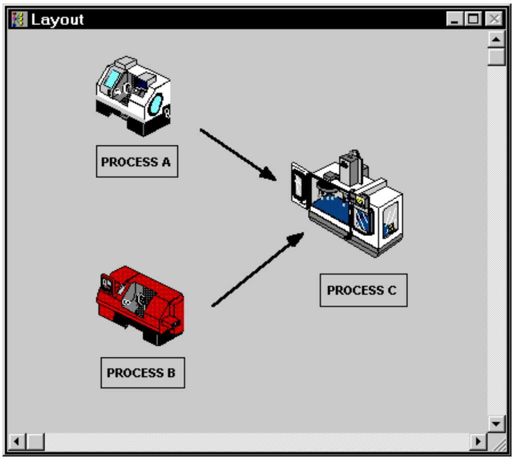
When using the ACCUM statement, make sure the capacity of the location where you are using the ACCUM statement is at least as large as the amount accumulating.

Modeling Priorities

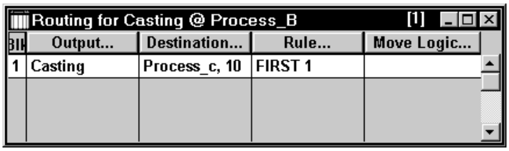
Priorities are an important part of modeling any system. They range from determining which location has priority for processing parts to choosing the appropriate resource. Priorities allow you to determine the order in which events occur in the simulation. The three most common uses of priorities deal with choosing processing destinations, selecting resources, and prioritizing downtimes.

Choosing a Processing Destination

When one downstream destination exists and there are two or more upstream entities competing to get there, you can use priorities. In the following example, two entities at different locations are both trying to get to Process C.



If you want the entity coming from Process B to have priority and go to Process C first, define a priority in the destination field for the routing of Process B. In the following example, we entered a priority of 10 for the routing of the entity Casting at the location Process_B.


Routing for Casting @ Process_B [1]

Blk	Output...	Destination...	Rule...	Move Logic...
1	Casting	Process_c, 10	FIRST 1	

How to enter a priority for a destination:

Enter a priority value directly in the destination field, making sure there is a comma separating the destination and priority.

or...

Click the **Destination...** button and enter the priority in the appropriate field.

Selecting Resources

Similar to the previous example, this type of prioritizing refers to which of two or more processes, requesting the same resource will have priority in capturing the resource. In the Operation field, in conjunction with the USE or GET statements, the process with the higher priority number gets the resource first.

Downtimes and Preemption

ProModel uses priorities in conjunction with resource or location downtimes to determine which defined downtime takes priority. Downtimes with priorities that are 99 or less do not interrupt or "preempt" the activity, but begin after the current process completes. Downtimes with a priority of 100 or higher preempt the current activity. After the downtime, the activity resumes from where it left off.

Displaying Statistics On Screen

Effective use of statistics lends meaning and credibility to simulation models. This section is designed to show ways of displaying statistics on screen while the simulation is running. It also provides examples you can incorporate into any model.

Displaying on-screen statistics is valuable for adding model clarity as well as providing a way to keep track of what is happening in the model. The technique used for displaying statistics is to define variables within your model, perform certain functions on those variables (e.g., addition, subtraction, etc.), and then display those variables on the screen.

ProModel defines variables as symbols (e.g., x, y, and var1) that hold numeric values. They are defined in the Variables edit table. When defining variables, you must include the following information:

- Name (may be alphanumeric combination)
- Type (real or integer)
- Initial value (may be any value)

After defining a variable, you may display it anywhere on the layout screen by simply clicking with your mouse in the desired position with the variable highlighted in the edit table.

How to show system throughput on the screen:

1. Define the variable "thruput."

2. Display the variable on the screen.

3. In the processing edit table, increment the variable by one each time an entity leaves the system.

Another common, on-screen statistic is work in process (WIP). To show this statistic on the screen, follow the first two steps above. To accurately indicate the WIP in your model, increment a variable as entities enter the system and use the decrement statement as they exit the system.

How to display the total system time for entities:

1. Assign the simulation clock time to an entity attribute at the beginning of your simulation.

2. Subtract that attribute from the current clock time at the ending location and assign it to a variable you can display.

For example, to display total system time for entities in the system, define a variable and an attribute. The entity attribute, Input_Time, records the time an entity entered the system. The variable, Sys_Time, records the elapsed time each entity spent in the system (clock-input time). By displaying the variable on the layout, you will have a dynamic value showing system time of the exiting entity.

Creating Pull Systems

Types of Pull Systems

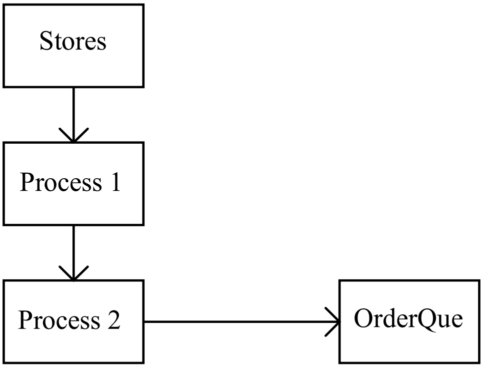
A pull system is a system in which locations produce parts only on downstream demand. There are two types of pull systems:

- those based on limited buffer or queue sizes.
- those based on more distant “downstream” demand.

The first method, that of limited buffer sizes, is quite easily to model using ProModel. By simply defining limited capacity locations, a preceding location will not send parts until capacity is available. This method works fine for most pull systems. The second method, triggering part movement, based on more distant downstream demand requires use of the SEND statement to trigger part movement.

There are additional ways to model pull systems using ProModel. As you review the modeling requirements of your own pull system, you should verify that these constructs will satisfy your needs.

This section shows how to use the SEND statement to model a pull system. The diagram below shows the pull system we will model. Orders for finished goods arrive at the OrderQue. The arriving order triggers the release of a unit from the location Stores. The order continues to wait at OrderQue until the unit goes through Processes 1 and 2. At Process_1, Unit processes for two minutes. At Process_2, Unit processes for four minutes. Finally, the Unit joins to the requesting order waiting in the OrderQue.



Creating the Pull System

Now let's examine the ProModel steps to build this pull system.

1. Define four locations: Stores, Process_1, Process_2, and OrderQue.
2. Define two entities: Order and Unit.
3. Define the processing as shown previously.
4. Define the arrivals. Schedule Units to arrive at location Stores. Schedule Orders to arrive at location OrderQue.
5. The entity Order “drives” the system by sending Units to Process_1 to fulfill the order. In the operation logic at OrderQue, use the SEND statement to send a Unit to Process_1 from location Stores. A corresponding SEND rule must be used as a Routing Rule for Unit at Stores.
6. After the SEND statement in the operation logic at the OrderQue, use a JOIN statement to join a Unit to an Order. A corresponding JOIN rule must be used as a Routing Rule for Unit at Process_2.
7. Place the processing times for the Unit at Process_1 and Process_2.

When you finish, the processing and routing should appear as shown below.

Process Table **Routing Table**

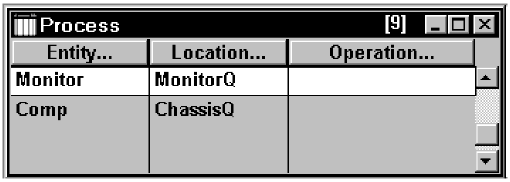
	Location	Operation (min)	Blk	Output	Destination	Rule	Move Logic
Unit	Stores		1	Unit	Process_1	SEND 1	
Unit	Process_1	WAIT 2	1	Unit	Process_2	FIRST 1	MOVE FOR .5
Unit	Process_2	WAIT 5	1	Unit	OrderQue	JOIN 1	MOVE FOR .5
Order	OrderQue	SEND 1 Unit TO Process_1 JOIN 1 Unit	1	Order	EXIT		

Making Assemblies

An assembly occurs when you attach specific items or entities to another entity such as a base part or frame. To assemble entities, use the JOIN statement. Implementing JOIN is a two-step process:

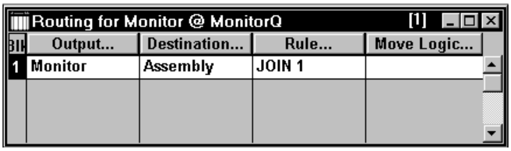
1. Employ the JOIN statement at the designated assembly location.
2. Use the JOIN routing rule for all joining entities.

It is helpful to designate one of the joining entities to be the "base" entity which issues the JOIN statement. In the following example, "Comp" is the base entity. All other joining entities must travel to the assembly location on an "If Join Request" routing rule. Note that for "Monitor" traveling to the assembly location, the word "JOIN" appears in the Rule field. This indicates that "Monitor" will go to the location Assembly only if a JOIN statement requests it elsewhere in the model.



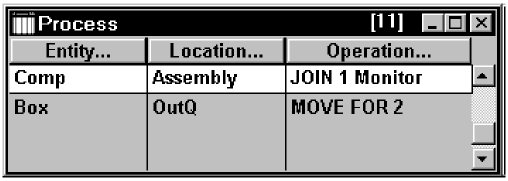
Process [9]

Entity...	Location...	Operation...
Monitor	MonitorQ	
Comp	ChassisQ	



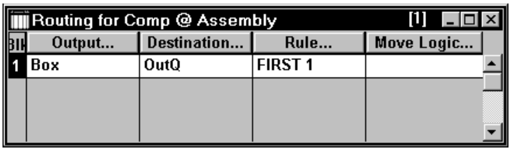
Routing for Monitor @ MonitorQ [1]

Blk	Output...	Destination...	Rule...	Move Logic...
1	Monitor	Assembly	JOIN 1	



Process [11]

Entity...	Location...	Operation...
Comp	Assembly	JOIN 1 Monitor
Box	OutQ	MOVE FOR 2



Routing for Comp @ Assembly [1]

Blk	Output...	Destination...	Rule...	Move Logic...
1	Box	OutQ	FIRST 1	

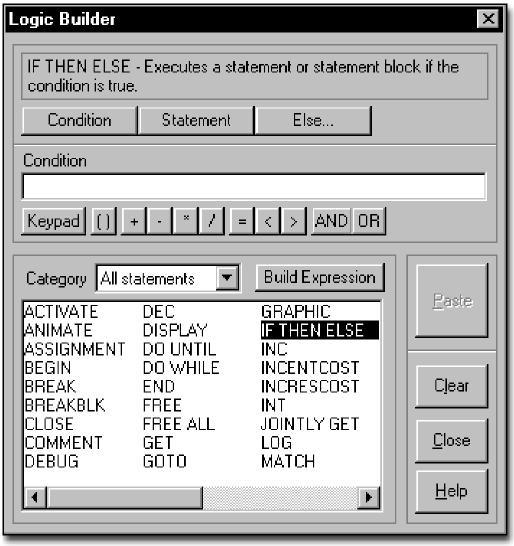
Several other ProModel statements similar to JOIN exist such as GROUP/UNGROUP, COMBINE, LOAD, and ACCUM. You generally use these statements for temporary or permanent batching. To learn more about these statements, see the discussion on Batching and Unbatching earlier in this section.

For more information see "Join" on page 504.

Chapter 7: Building the Logic

Logic Builder

The Logic Builder provides a quick and powerful way to create valid statements and expressions in logic windows or fields. It takes you through the process of creating statements or expressions, as well as providing point-and-click access to every element defined in your model. The Logic Builder knows the syntax of every statement and function, and allows you to define logic simply by filling in the blanks.



IF THEN ELSE - Executes a statement or statement block if the condition is true.

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.

Using the Logic Builder

When the Logic Builder is opened from a logic window, it remains on the screen until you click the Close button or close the logic window or table from which it was invoked. This allows you to enter multiple statements in a logic window and even move around to other logic windows without having to constantly close and re-open the Logic Builder. The Logic Builder closes automatically when pasting to an expression field.

You can move to another logic window or field while the Logic Builder is still up by right clicking in that field or logic window. The Logic Builder is then reset with only valid statements and elements for that field or window, and will paste the logic you build into that field or window.

How to build a statement or expression:

1. Right click in an expression field or logic window to open the Logic Builder or click on the Build button in a logic window.

2. Select the statement or expression you want to use from the list box. When opened from a logic window, you have the option to click on the Build Expression button to create only an expression.

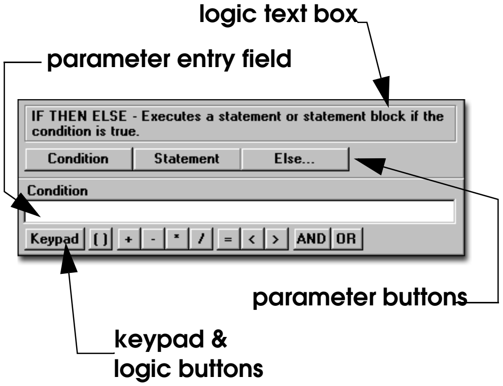
3. Enter the parameters for the statement or expression. These may be expressions using model elements and/or functions or other statements. Parameters may be edited or entered manually in the Parameter Entry field.

4. Paste the results into the logic field or window by clicking the Paste button.

Logic Builder Components

When invoking the Logic Builder from a logic window, you have the option of building either statements or expressions. Different buttons and lists appear in the Logic Builder as you use the Logic Builder's options depending upon whether you are selecting a statement or building an expression. The Logic Builder shown at the beginning of this section shows a statement being selected for building.

At the top of the Logic Builder is a display (logic text box) of the statement or expression you are building exactly as it will appear after it is pasted into the logic window. A brief description of the selected statement or function is displayed in the logic text box. This description is replaced with the statement or function syntax when you type a parameter, click a parameter or logic button, or double-click on the statement name. Other components of the Logic Builder are as follows:

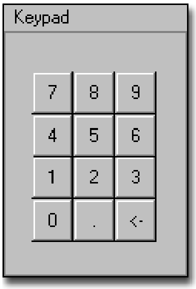


IF THEN ELSE - Executes a statement or statement block if the condition is true.

Parameter buttons Below the logic text box are one or more buttons to control which parameter to enter for the statement or expression. Parameters can be expressions, statements, or functions. These buttons only appear when parameters may be required by the statement, and may change when you select a different statement. The name of the currently selected button appears immediately below the row of buttons and indicates whether or not the parameter is optional.

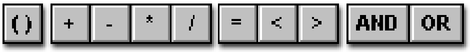
Parameter entry field This edit field allows you to enter or edit the current parameter. This only appears when parameters may be required by the statement. As soon as something is entered in this field, the Logic Builder switches to build mode to allow selection of functions or elements of the model.

Keypad button Click on this button to display the numeric keypad for entering numbers in the parameter entry field without using the keyboard.

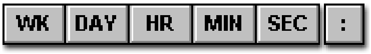


Logic buttons These buttons can be used to insert logical operators and other punctuation in the parameter entry field above. When you click the button, the operator is inserted at the cursor position in the field. A button appears only when the currently selected parameter can use that particular logical operator.

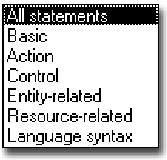
- Logical & String Operators:



- Time Unit Operators:



Category This combo box allows you to select which type of statements appear in the statement selection list below it. You can select all or a specific type.



Build Expression button This button allows you to create only an expression. It displays the logic elements list (see below) for you to create the expression. An expression consists of a combination of numbers, model elements, and/or functions, but does not include statements.

Statement selection list Choose which statement you wish to use from this list. Only valid statements are displayed for the logic window or field you are using.

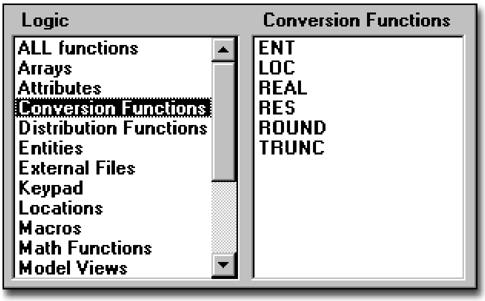
Paste button This button pastes the text of the logic text box into your logic window or field. It is only available once the minimum requirements of the statement or expression have been completed. The Paste button closes the dialog if you are pasting to a field.

Clear button This button clears whatever you have done since the last paste and allows you to start over.

Close button Closes the Logic Builder without pasting the current logic text box.

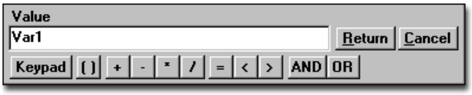
Help button Launches the context sensitive help system.

Logic Elements When editing an expression in the parameter entry field, the Statement selection list is replaced by Logic Elements. The box on the left lists logic and model element types. The box on the right lists individual selections from the logic or model element type selected.



Return and Cancel Buttons When editing the parameters of a function or nested statement, two additional buttons appear to the right of the parameter edit box: Return and Cancel.

- **Return button** This button (available only when required parameters have been entered) returns to the previous parameter entry field and pastes the function or statement at the last cursor position.



Value: Var1

- **Cancel button** Aborts editing of the function or nested statement and returns to editing the previous statement or function.

Selecting Statements

The first thing to do in creating a statement with the Logic Builder is to select the desired statement from the statements list box. You can restrict the list of statements to choose from using the Statement Type combo box above the statement list. If you are just starting to use ProModel, you may want to select the Basic Statements type to list only the most commonly used statements.

To select a statement, left click on the statement name in the list box. The statement name appears in the logic text box along with a brief description. The parameter buttons also appear just below the logic text box.

Before you begin editing the parameters of the statement, you can select a different statement. However, once you begin defining a parameter, you must click the Clear button to abandon that statement and select another.

Editing Statement Parameters

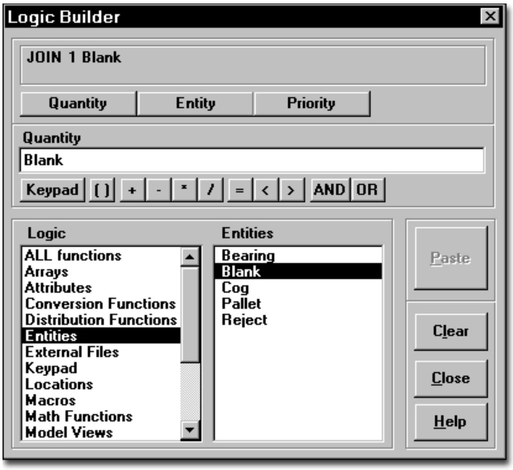
Building a statement is simply a matter of filling in the parameters. To enter a parameter, click the appropriate parameter button below the logic text box. The parameter name is displayed above the parameter entry field. Whatever you type in this field or select from the Logic Elements list replaces the parameter name in the logic text box. The parameter name reappears when the entry field is cleared. The names of optional parameters are not displayed in the logic text box.

An optional shortcut to begin editing the statement's first parameter is to double click on the statement name in the statement list box.

Selecting Logic Elements

The Logic Elements list box, containing model elements and functions, appears with a selection list box to its right. A number pad element is included in the list, which can also be accessed using the Keypad button.

When you click on an item in the Logic Elements list, the model elements or functions related to that item are listed in the selection box on the right. For example, when you click on Entities, the selection list is entitled Entities and it contains all the entities defined in the model.



JOIN 1 Blank

Quantity: Blank

Logic: Entities

Entities: Blank

How to place a model element in the parameter entry field:

1. Left click on the desired element type in the Logic Elements list box. The elements for that type will be placed in the selection list box.

2. Left click on the desired element to paste it into the parameter entry field at its current cursor position. Note that the element is highlighted in the parameter field; clicking on another element will replace it.

How to place a function in the parameter entry field:

1. Left click on the desired function type or on All functions in the Logic Elements list box. The functions for that type will be placed in the selection list box.

2. Left click on the desired function to paste it into the parameter entry field at its current cursor position. The Logic Builder jumps into build mode for that function's parameter(s). Note that two new buttons are placed to the right of the parameter edit field: Return and Cancel.

3. To fill in the function's parameter(s), repeat steps one and two. If you do not want to use this function, abandon it by clicking on the Cancel button.

4. Once the function's parameters are complete, click on the Return button. The parameters you just completed will be pasted into the function's parameter entry field, and the completed function with its parameters is pasted into the original statement's parameter entry field.

Nested Functions & Statements

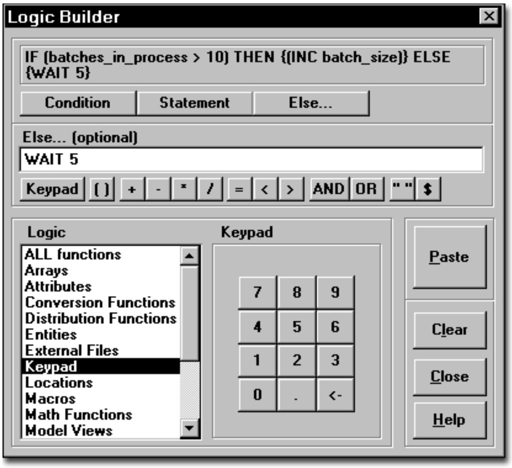
When you select a function as the parameter for your statement, you must also define that function's parameter(s). In defining the function's parameter(s), you may use another function which will also require defined parameters. This second function is called a nested function. In addition, a function may be nested within a nested function. Functions can be nested as many levels as you like. In this way, the logic builder helps you create complex expressions that would be difficult to enter manually.

Control statements such as IF...THEN and WHILE...DO contain parameters that are themselves statements. These are called nested statements. For example, an IF...THEN...ELSE statement might look something like this:

Nested IF, THEN, ELSE

```
If (Variable1 > 10) Then
    {
    < Statement1 >
    < Statement2 >
    }
Else
    {
    < Statement3 >
    < Statement4 >
    }
```

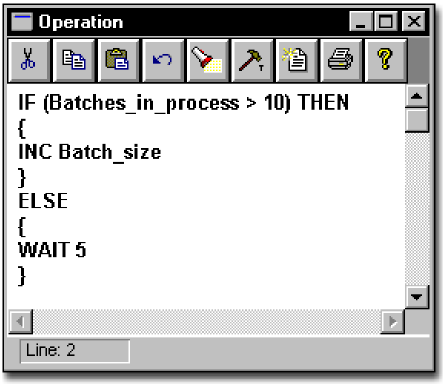
One or more statements may appear in the block between the curly braces. The Logic Builder allows you to define the first statement of the block. To add other statements to the block, place the cursor in the logic edit window where you want the next statement and use the Logic Builder to write the statement and paste it in. For example, in the following Logic Builder window, an IF...THEN...ELSE statement is being built.



```
IF (batches_in_process > 10) THEN {(INC batch_size)} ELSE
{WAIT 5}
```

Else... (optional): WAIT 5

The ELSE statement is built with the parameter button labeled Else and the statement following THEN is built using the button labeled Statement. When you click on the parameter button labeled Statement, the statement list box is displayed where you can select from the list of valid logic statements. Click on paste in the above example and you get:



```
IF (Batches_in_process > 10) THEN
{
INC Batch_size
}
ELSE
{
WAIT 5
}
```

The parameters of nested statements may be model elements or functions, and within those functions you may have nested functions. This allows you to easily build complex control statements without worrying about syntax and placement of nested statements and functions.

Creating Expressions or Pasting Logic/Model Elements Only

In addition to creating statements, the Logic Builder can also be used to create just an expression or to simply paste a particular element such as a variable or resource name. You may not need to create a complete statement, or the field may not accept statements. Pressing the Build Expression button allows you to build and paste expressions, including individual logic or model element names, into your logic window or field.

The expression being built or element being selected is displayed at the top of the window in the logic text box. Use the parameter edit field to build the expression. You can use model elements and/or functions in your expression. When you are finished, click the Paste button to place the expression or selected logic/model element in the logic window or edit field.

Operation Logic

Operation logic defines what happens to an entity when it enters a location. Operation logic is optional, but typically contains at least a WAIT statement for the amount of time the entity should spend at the location. For modeling purposes, the exact nature of the operation (joining, grouping, etc.) is irrelevant. What is essential is to know what happens in terms of the time consumed, the resources used, and any other logic that impacts system performance. For operations requiring more than a time and resource designation, detailed logic may need to be defined using IF...THEN or action statements.

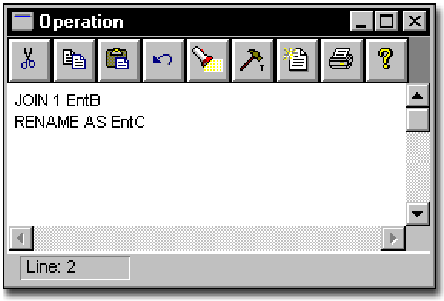
Special operation statements are provided to define the activities that are to occur. By using operation logic, any of the following activities can be defined:

- Detain an entity for a specified length of time while an activity is performed.
- Detain an entity until one or more resources are obtained.
- Detain an entity until one or more additional entities are joined to it.
- Detach one or more entities from the current entity.
- Consolidate one or more entities into a group.
- Separate an entity into two or more entities.
- Detain an entity until a particular system condition is reached.
- Destroy an entity.
- Create one or more new entities.
- Execute a block of logic (assignment of variables, etc.).
- Signal the start of some other action in the system (e.g., place an order).
- Make some decision about further 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 statements, functions, and distributions available in the operation field are discussed in detail, including examples, in “Statements and Functions” on page 439.

Example

Each entity processes the operation statements defined for it at a particular location, independent of other operations performed on other entities at the same location. The following example presents the operation logic for an entity joining an entity, EntB, and renaming the entity as EntC.



```
JOIN 1 EntB
RENAME AS EntC
```

Preemption Process Logic

With Preemption Process Logic, ProModel makes it possible to control preemption rather than limit you to default preemption priority levels and values. This feature pertains only to the preemption of entities using a location. It does not include preemption of downtimes.

Normally, if an entity or downtime attempts to preempt an entity that is using a location, the location is immediately preempted (assuming it is preemptable). Once the location finishes the preempting activity, the original preempted entity regains possession of the location and resumes processing where it left off.

To override this default preemption procedure, in the case of entities that are preempted by another entity or a downtime, ProModel allows a Preemption Process Record to be defined which postpones the actual preemption of the location until after the current entity explicitly releases it.

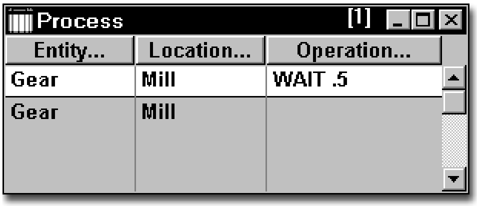
The Preemption Process Record, allows you to control if and when the location is given up. In the case of a preemptive request for a location, the preemption process record allows the entity to be routed elsewhere if desired. If a preemption process is defined, the actual preemption does not occur until the preemption process explicitly frees the location.

After the location is given up, the entity may (1) elect to use an alternative location to complete the process, or (2) seek to regain access to the same location to complete the process.

While an entity is executing a preemption process, it cannot be preempted by any other entity or downtime.

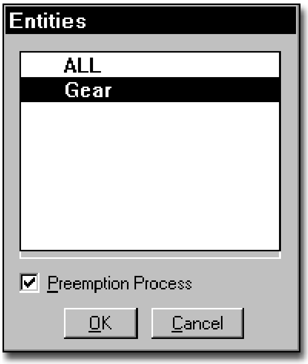
How to create a preemption process record:

1. Using the same entity and location names, create a process record somewhere following the process record where the preemption may occur.



Entity...	Location...	Operation...
Gear	Mill	WAIT .5
Gear	Mill	

2. Click on the **Entity** button to display the Entities dialog shown here.



☑ Preemption Process

3. Check the **Preemption Process** option box and click OK.

4. Click on the Operation button to enter the preemption logic. You can use any valid operation logic including delays. It is recommended that you enter a comment as the first line in the logic indicating that this is a preemption process. This will make the record easily identifiable as a preemption process.

Please note

When a preemption occurs, the entity looks forward and then from the beginning of the process list trying to find a preemption process that matches the same entity and location (a process with ALL as the entity name will match any entity). If a match is found, the preemption process gets executed. Otherwise the default preemption occurs. Only the first preemption process encountered will be executed in the event that multiple preemption processes are defined with the same entity and location names.

Possible Effects of Delayed Preemption

Several circumstances can be created through the use of preemption process records. An entity delaying a preemption, for example, may find at the conclusion of the delay that the preemption is no longer required, or that it faces an even higher preemption priority.

In cases where an entity has multiple locations or resources from which to choose, a preemptive request for any one of them is not necessarily a commitment to select that particular location. If any of the alternative locations becomes available, the entity will select it and withdraw the preemption request.

Functions for Defining Logic in a Preemption Process

The Preemption Process Logic feature includes the following functions for use in the preemption logic.

PREEMPTOR() Identifies whether a downtime or entity is making the preemptive request. It returns a 0 if the preemptor is a downtime, otherwise it returns the index number of the preempting entity.

TIMELEFT() Returns the amount of time remaining if the preemption occurred during a WAIT or USE statement. It returns a time value in default time units (real). If multiple entities are preempted from a location, it returns the longest remaining time for all of the entities.

Please note

The values returned by these functions must be checked before any processing delay occurs since they are updated whenever a preemption takes place. If the values must be referred to later, they should be assigned to the entity's attribute or to a local variable.

Preemption Process Example

In this example, a Gear may be preempted in its occupancy of the Lathe. This preemption may be because of either a preempting entity or downtime. Before the actual preemption takes place, the operation time for the Gear is interrupted and the Gear immediately begins processing the operation logic defined in the preemption process.

In the preemption process, the remaining operation time is stored in Attr1. The Gear routes to Lathe_Backup where it finishes processing. Because the backup lathe is not as efficient as the other Lathe, it takes 50% longer to process the Gear on Lathe_Backup. Therefore, we multiply the time left to process the Gear by a factor of 1.5.

Process Table

	Location	Operation (min)
Gear	Lathe	WAIT 10 min
Gear	Lathe	Attr1=TIMELEFT()
Gear	Lathe_Backup	WAIT Attr1*1.5

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Gear	Mill	FIRST 1	
1	Gear	Lathe_Backup	FIRST 1	
1	Gear	Mill	FIRST 1	

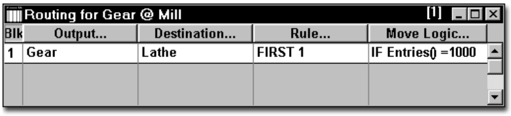
Routing Move Logic

To accommodate the use of multiple resources for entity movement, the Move Logic window allows you to define the method of movement as well as any other logic to be executed upon movement.

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, or MOVE WITH) is executed or until 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 a 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 used to transport the entity.

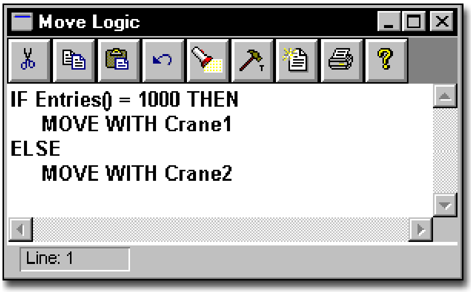
When you access the Processing module in the Build menu, the Routing edit table appears with the Move Logic button in the right hand column as shown here.



Routing for Gear @ Mill [1]

Blk	Output...	Destination...	Rule...	Move Logic...
1	Gear	Lathe	FIRST 1	IF Entries() =1000

When you click the Move Logic the following Move Logic window appears.



```
IF Entries() = 1000 THEN
   MOVE WITH Crane1
ELSE
   MOVE WITH Crane2
```

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.

Move-Related Statements

Admissible statements in the Move Logic window include the new move-related statements listed here. A brief description of how each statement functions in ProModel follows the list. See "Statements and Functions" on page 439 for complete syntax, description, and examples of these statements.

MOVE statements

MOVE FOR <time>

MOVE WITH<res1>{,{p1}} {FOR <time>} {THEN FREE}

MOVE ON <path network>

MOVE FOR <time> Used to specify the amount of time required to move the entity. If the <time> is zero, events for other entities occurring at the same simulation time will be processed before any additional logic is processed for the current entity.

MOVE WITH This statement is used to move an entity using a designated resource, such as a forklift. With the OR operator, you can designate alternative resources for making the move. In this case, the statement captures the first available alternative resource designated in the expression and makes the move. If one of the resources is already owned by the entity, that resource will be used. Please note that you cannot use the AND operator to capture (and move with) more than one resource with this statement. To move an entity with multiple resources, you must use a GET statement to capture the additional resources.

This statement also allows you to set the priority (p1) for accessing the designated resource. If the resource is already owned by the entity, this priority is ignored.

If the resource is static, a time (FOR <time>) may be specified for the move.

The resource used to make the move is only freed if the THEN FREE option is used.

MOVE ON <path network> Use this statement to move an entity along a path network.

Please note

Not entering no move related statement in the Move Logic window causes entities to move immediately to the next location when the move logic is completed and begin executing the operation logic for that location.

Related Logic Statements

In addition to these routing specific statements, all statements and functions allowed in exit logic may be used in the Move Logic window. Note, however, that the LOCATION() function returns the *current* location when executed *before* the MOVE statement and returns the *destination* location when executed *after* the MOVE statement. Additional statements permitted before a move-related statement include: WAIT, WAIT UNTIL, GET, JOINTLY GET, and USE.

See “Statements and Functions” on page 439 for full description, elements explained, and examples.

Statement Processing

Statements executed before a MOVE related statement are processed after the entity claims the next location but before the entity actually departs from the current location.

Statements executed after a MOVE related statement are processed after the move has been completed but before the entity enters the location. If there is no move logic, the entity continues processing until it encounters an implicit WAIT. However, if “MOVE FOR 0” is placed in the move logic, the event list is broken and other processes scheduled to occur at the same time are executed. Once these processes are executed, the entity enters the destination location.

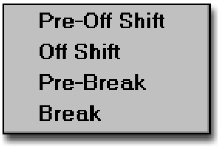
This processing sequence allows you to GET or JOINTLY GET one or more resources before the move and optionally FREE one or more resources at the end of the move. Please note that if a move related statement is not encountered in the logic, an implied MOVE will be assumed and executed at the end of the logic.

Shift & Break Logic

Shift & Break Logic

Shift and break logic are optional and are defined in four distinct logic windows. Each logic window is executed in a specific sequence throughout the simulation run. You can define logic that controls how resources and locations go off duty or off line and what happens once they are offline.

To define shift or break logic, click on the Logic button in shift assignments to display a submenu of four shift events 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 *Sequence of Events* below.



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, and 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-start logic when speaking of either one.)

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

How to determine the 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-start logic for that particular location or resource is executed.

2. After executing the pre-start 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.

Please 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 several functions and statements specifically for off-shift and break logic: SKIP, PRIORITY, DTLEFT(), FORLOCATION(), FORRESOURCE(), and RESOURCE(). Following is a brief description of each. For more details, see “Statements and Functions” on page 439.

SKIP If used in pre-start logic, it causes the off-shift or break time (including any off-shift or break logic) to be skipped so that the 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 file to be skipped. This allows you to specify a WAIT statement for the off-line time and then SKIP the off-line time defined in the shift file.

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 if any preemption may occur at that time. This statement is not allowed in pre-off shift or pre-break 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 that 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 fol0lowed 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.

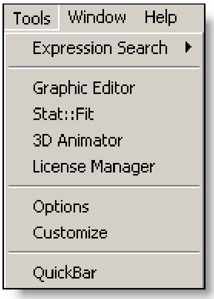
RESOURCE() This returns the name-index number of the resource currently processing the off-shift or break logic.

In addition to these functions, the LOCATION() and DTDELAY() functions are particularly useful when defining off-shift and break logic.

Chapter 8: Using Auxiliary Tools

Tools Menu

The Tools menu gives you access to powerful tools to help you through the model building process.



Expression Search Allows you to perform global search and replace functions on expressions throughout any part of the model.

Graphic Editor Allows you to create, edit, rearrange, or delete library graphics for use as entities, locations, resources and background graphics.

Stat::Fit Launches Stat::Fit and allows you to fit analytical distributions to user data.

3D Animator Launches 3D Animator if it is installed. If 3D Animator is not installed, a dialog will provide information about purchasing 3D Animator or obtaining a demonstration version.

License Manager Launches the license manager. Since license changes cannot be made while ProModel is running, you will be prompted to close ProModel prior to running the License Manager.

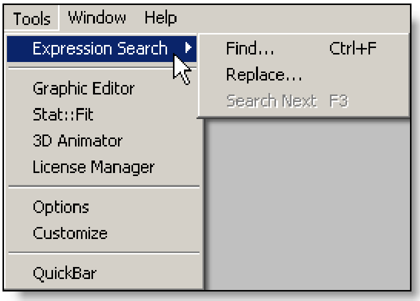
Options Allows you to set various directory and display defaults.

Customize Allows you to customize the Tools menu.

Additional Tools Tools can be added to the Tools menu by using the Customize option. These tools are discussed in the latter part of this chapter.

Expression Search

The Expression Search feature is used to find or replace text entered into logic windows and expression fields, such as location downtime logic or location capacity. Name fields can be found, but not replaced. Reference fields can be found and replaced. However, when the name of a location, resource, or entity is changed, the user will be prompted to automatically change all references to the new name. There are three types of searches: Find expression, Replace expression, and Search Next expression.



How to perform an expression search:

- Select **Expression Search** from the **Tools** menu.

Expression Search Sub-Menu Choices

Find

A dialog box gives you the following options after choosing Find from the Expression Search submenu:

Modules to Search Check the modules you want to search. “Other” includes all other edit tables and some dialog boxes where text is entered for defining the model, such as attributes or arrays.

Search Notes Check this box to include Notes fields in the text search.

Whole Words Only Check this box to search for only whole words or groups of whole words that match the text to find. For example, searching for “Attr” without the Whole Words Only box checked will find “Attr1” and “Attr2” whereas a search with the box checked would find neither.

Text to Find Enter the text expression you want to find.

Replace

Choosing replace gives you these options in addition to the find options:

Prompt on Replace Check this box if each time ProModel finds a match, you want ProModel to ask if you want that particular match changed to the replacement text. The prompt will give you the option to replace that particular match, skip that particular match, or to cancel the search altogether.

New Text Enter the text you want to replace the search text.

Search Next

Choosing Search Next in the Expression Search submenu will resume the most recently canceled search. For example, suppose you begin a search and then break out of the search to adjust something in the model. If you want to continue the original search, you can select Search Next and ProModel will start the search again at the place you stopped searching.

Find Expression

The Find Expression option allows you to find each occurrence of an expression in a model.

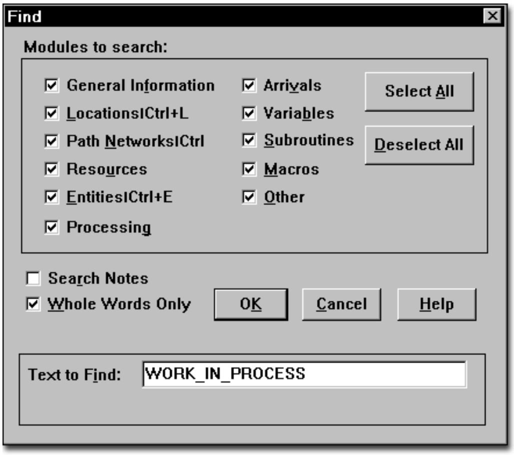
How to find an expression:

1. From the **Tools** menu, select **Expression Search**.

2. Select **Find...** from the submenu.

3. Supply the necessary details in the Find dialog box shown below. Clicking Select All or Deselect All button will check or uncheck every module.

4. Click **OK**.



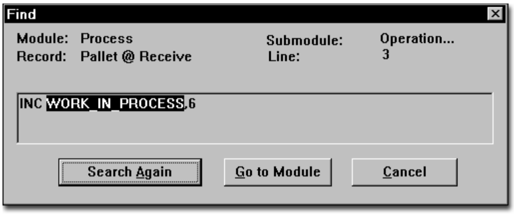
Modules to search:

☑ General Information
☑ Locations|Ctrl+L
☑ Path Networks|Ctrl
☑ Resources
☑ Entities|Ctrl+E
☑ Processing
☑ Arrivals
☑ Variables
☑ Subroutines
☑ Macros
☑ Other

□ Search Notes
☑ Whole Words Only

Text to Find: WORK_IN_PROCESS

Once the Expression Search has found the first occurrence of an expression, a dialog box will appear giving information on exactly where the text was found. Then ProModel will display the following dialog, including a box displaying the entire line on which the text was found, with the search expression highlighted.



Module: Process
Record: Pallet @ Receive
Submodule: Operation...
Line: 3

INC WORK_IN_PROCESS,6

How to find the next match of a text expression:

- Click on **Search Again**

Replace Expression

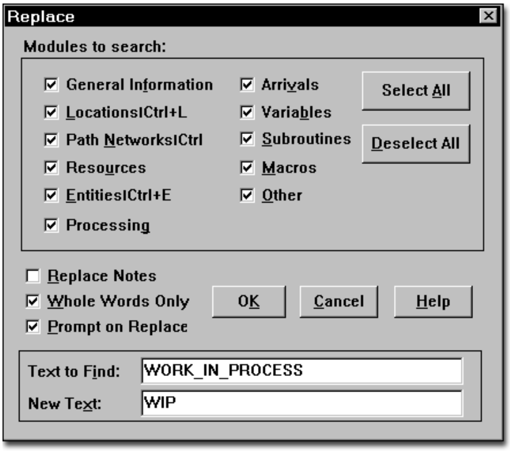
The Replace Expression option allows you to find each occurrence of an expression in a model and replace that expression with a new expression.

How to replace an expression with another expression:

1. Select **Replace...** from the Expression Search submenu.

2. Supply the necessary details in the Replace dialog box shown below. Clicking

Select All or **Deselect All** button will check or uncheck every module.



Replace

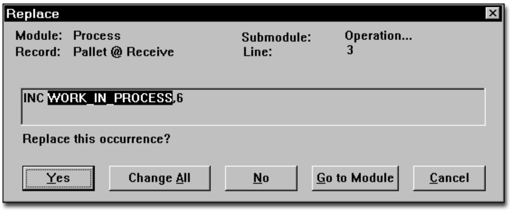
Modules to search:

☑ General Information
☑ LocationsICtrl+L
☑ Path NetworksICtrl
☑ Resources
☑ EntitiesICtrl+E
☑ Processing
☑ Arrivals
☑ Variables
☑ Subroutines
☑ Macros
☑ Other

□ Replace Notes
☑ Whole Words Only
☑ Prompt on Replace

Text to Find: WORK_IN_PROCESS
New Text: WIP

3. Click on **OK**. If you have chosen "Prompt on Replace," ProModel will then display the following dialog box if it finds the text you specified.



Replace

Module: Process
Record: Pallet @ Receive
Submodule: Operation...
Line: 3

INC WORK_IN_PROCESS,6

Replace this occurrence?

4. Choose:

Yes to change the text and search for the next match.

Change All to change every match.

No to skip this match and search for the next match.

Go to Module to edit the text directly.

Cancel to leave the match intact and stop searching.

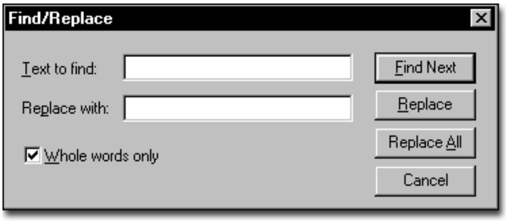
Important Notes Regarding Expression Searches

1. Not every field of every module is included. Fields such as statistics, text in graphics, or yes/no fields which may not be edited cannot be searched for or replaced. To replace record identifiers, see number six.
2. Under "Modules to Search," the Other option refers to information entered in places not listed in the dialog box, including the Simulation Options dialog box.
3. Notes fields are not part of the actual model data, therefore they are not automatically included in the search. Notes also include comments in the model. If you want to search or replace Notes fields, check the Search Notes option in the Find or Replace dialog boxes.
4. The Whole Words Only option interprets words loosely enough to distinguish words not separated by spaces. For example, searching for "Attr1" in the expression Attr1=Attr1+Attr2, would find both occurrences. You can search for expressions longer than a whole word, such as "Attr1=Attr1+Attr2," as whole word expressions. To find a portion of a name, like "Attr" in "Attr1," deselect the Whole Words Only option.
5. Once the Expression Search has found the first occurrence of an expression, a dialog box will appear giving information on exactly where the text was found. This includes a box displaying the entire line on which the text was found, with the search expression highlighted. In some cases the box may be too small to display the entire line. To see the hidden portion of the text, left click with the left mouse button on the text, and use the left and right arrow keys to scroll the text horizontally.

6. The Replace feature cannot be used to change an element identifier. To change all occurrences of a model element name (such as a location name), change the name of the element where it is defined and all other expressions containing the name, as well as any references to this record, will be changed automatically.

Local Find and Replace

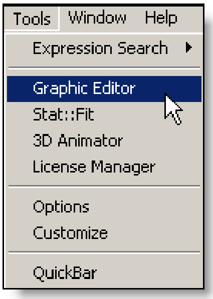
In the process of creating logic, you may need to search for specific text. The local find and replace button in the operation logic window opens a dialog that will allow you to search the logic for specific text.



☑ Whole words only

Graphic Editor

The Graphic Editor allows you to create, edit, rearrange, or delete library graphics within a particular graphics library file. You can also copy graphics from one library to another. Graphics from several libraries can even be merged into a single graphic. Each graphics library is saved with the grid size and scaled used to create the graphics.



How to access the graphic editor from within ProModel:

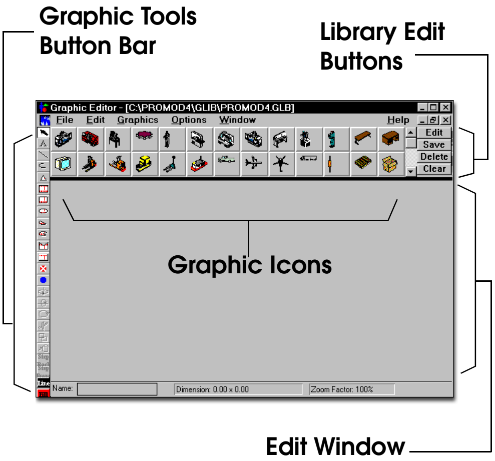
- Select **Graphic Editor** from the **Tools** menu.

How to access the graphic editor from the program manager:

- Double click the Graphic Editor icon in the ProModel Group. This method allows you to use the Graphic Editor as a separate application from ProModel.

Overview

The Graphic Editor consists of a menu bar, a Graphic Tools button bar, a Graphic Library menu, Library Edit buttons, an Edit window, and a set of icons representing the graphics in the graphics library.



Each of the tools in the Graphic Tools button bar is discussed in this section, including the procedures for creating and editing a graphic's components (called "objects") using the drawing tools.

Graphic Editor Menus

The Graphic Editor menu bar includes the following menus:

File The File menu allows you to open a graphics library for editing and saving a current library. It also allows you to print a single graphic or an entire library.

Edit The Edit menu provides functions for selecting and duplicating one or more objects that comprise a library graphic. In addition, it provides functions to import and export graphics from other applications.

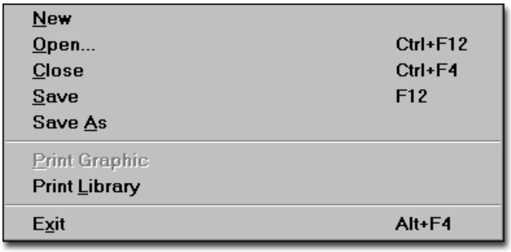
Graphics The Graphics menu is for manipulating one or more objects that comprise a library graphic. With this menu, you can group several objects together, flip and rotate objects, and alter the color, fill pattern and line style of objects.

You can also adjust the dimensions of the entire graphic.

Options The Options menu controls the editing environment. With this menu you can use a grid to help you align component objects, edit that grid, and require objects to snap to it. Finally, the Options menu allows you to zoom in and out on the graphic so you can edit the graphic at different sizes.

Window The Window menu allows you to arrange the windows (or iconized windows) currently displayed on the screen such that all windows are visible at once. It also allows you to bring any individual window to the forefront of the display. This is particularly useful when you are opening multiple graphic libraries and want to view all libraries simultaneously.

File Menu



New Creates an empty graphics library.

Open Brings up the Open Library Graphics dialog box for specifying which graphics library file to retrieve. Graphics library files have the file extension GLB.

Close Closes the current graphic library. If the graphics library has been changed since the last save, you will have the option to save it.

Save Saves an open graphics library under the current file name or prompts you for a name if the graphics library has not been named.

Save As Brings up the Save As dialog box for saving the current Graphics library file under a new filename. Graphic Library files have the file extension GLB.

Print Graphic Prints the graphic in the Edit window only.

Print Library Prints the entire current graphic library.

Exit Quits the Graphic Editor with an option to save the current library if changes have been made since the last save.

Recently Opened Files Lists the five most recently retrieved graphics libraries. Selecting one of these options will retrieve the listed graphics library.

Opening a Graphics Library File

All individual library graphics are loaded from and saved to the current graphics library, which, by default, is the one specified for the current model. However, other graphic libraries may be opened for editing at any time. The name of the current library is displayed in the title bar of the window for each library. More than one library can be opened and viewed on the screen at a time. Opening more than one graphic library simultaneously facilitates copying graphics between libraries.

How to open another graphics library file:

1. Choose **Open** from the **File** menu.
2. Enter or select the name of the desired graphics library.
3. To view all open graphic libraries, choose **Tile** or **Cascade** from the **Window** menu.

Please note

A history list is given at the bottom of the File menu so you are able to quickly retrieve the last five libraries opened.

Closing a Graphics Library File

When you are finished working with a graphics library, you can close it to save screen space and memory. This option will not affect the graphic library used with any model.

How to close a graphics library file:

- Choose **Close** from the **File** menu.

Saving a Graphics Library File

Once a graphic has been created or edited and placed in the current library, the library file must be saved in order to make the changes permanent.

How to save a graphics library file:

- Select **Save** from the **File** menu to save the library with the same name.

How to save a graphics library file with a new name:

- Select **Save As** from the **File** menu to save the library with a new name.

Printing an Individual Graphic

How to print an individual graphic:

1. Double click the mouse on the desired graphic's icon, or select the graphic's icon and click the **Edit** button.

2. Select **Print Graphic** from the **File** menu.

3. Choose the desired options from the resulting Print dialog box and click **OK**.

Printing an Entire Graphics Library

How to print an entire graphics library:

1. Select **Print Library** from the **File** menu.

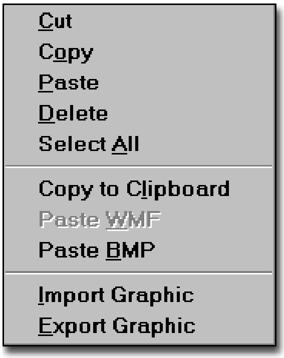
2. Choose the desired print options from the Library Print dialog box and select **OK**.

Please note

If more than one graphic library is open, ProModel will print the active graphic library only.

Edit Menu

Use the Edit menu for selecting and duplicating the individual objects comprising a library graphic. You may also use it to exchange graphics with other applications. To use the Edit menu functions, load the graphic you wish to edit by selecting its icon from the library and clicking the Edit button, or double click the icon.



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) to be pasted later.

Paste Adds the most recently cut or copied object(s) to the current graphic.

Delete Deletes the selected objects from the current graphic.

Select All Selects all of the objects comprising the current graphic.

Copy to Clipboard Copies the entire graphic to the clipboard as a bitmap so it can be pasted into other applications including word processors.

Paste WMF Pastes a Windows metafile (WMF) from the Windows clipboard into the Edit window. You must have previously copied a Windows metafile to the Windows clipboard in another application.

Paste BMP Pastes a bitmap file (BMP) from the Windows clipboard into the Edit window. You must first copy a bitmap file to the Windows clipboard.

Import Graphic Imports a WMF, BMP, PCX or GIF file into the Edit window.

Export Graphic Exports the graphic in the Edit window to a WMF or BMP file.

Importing a Graphic

How to import a graphic into a graphic library:

1. Select the box to which you would like to add the graphic in the library. If you want to create a new graphic, choose the blank box at the end.

2. Select **Import Graphic** from the **Edit** menu.

3. Enter the name of the graphic you would like to import.

4. Select **OK** to close the import graphic dialog box.

5. Click on the **Save** button on the top right side of the library window.

Exporting a Graphic

How to export a graphic:

1. Double-click the mouse on the desired graphic's icon, or select the graphic's icon and click the Edit button.

2. Select **Export Graphic** from the **Edit** menu.

3. Enter a valid DOS name for the graphic in the resulting dialog box.

4. Click the **OK** button in the export graphic dialog box.

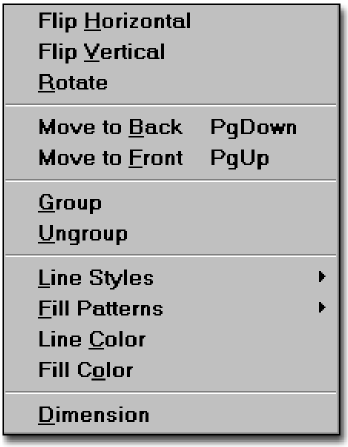
Copying a Graphic from One Library to Another

How to copy an icon from one library to another

1. Open both libraries.
2. Drag the graphic's icon from the first graphic library to the second graphic library (preferably left to right).
3. Save the destination library by choosing **Save** from the **File** menu.

Graphics Menu

The Graphics menu is for manipulating one or more objects that comprise a library graphic. With this menu, you can group several objects together, flip and rotate objects, and alter the color, fill pattern and line style of objects. You can also adjust the dimensions of the entire graphic.



Flip Horizontal Horizontally flips the entire graphic or any selected objects of the current graphic. This menu item works like the button of the same name described later in this chapter.

Flip Vertical Vertically flips the entire graphic or any selected objects of the current graphic. This menu item works like the button of the same name described later in this chapter.

Rotate Rotates the entire graphic or any selected objects of the current graphic 90 degrees clockwise. This menu item works like the button of the same name described later in this chapter.

Move to Front Moves the selected object in front of all other objects. Use this option to see an object obscured by other objects. This menu item works like the button of the same name described later in this chapter (to move an object one position forward at a time, use the Graphic Tools).

Move to Back Moves the selected object behind all other objects. Use this option to send an object obscuring other objects to the background. This menu item works like the button of the same name described later in this chapter.

Group Combines or groups several objects into a single object for sizing and editing.

Ungroup Ungroups several grouped objects so they may be edited individually.

Line Styles Allows the user to choose the line style including solid, dashed, line thickness, and optional arrowheads on either end of the line.

Fill Patterns Allows the user to choose the fill pattern for solid objects including transparent, slant, backward slant, grid, crosshatch, vertical, horizontal, solid, vertical gradient, and horizontal gradient.

Line Color Allows the user to choose the line color and create custom colors.

Fill Color Allows the user to choose the fill color and create custom colors for solid objects.

Dimension Brings up the Dimensions dialog box for defining the graphic dimension. The dimension can be height or width. The units can be feet or meters.

Please note

Line Styles, Fill Patterns, Line Color, and Fill Color set the feature and cause the setting to be applied to the currently selected elements.

Group

When using the graphic tools to create an icon, it is often helpful to group several graphics into a single graphic for editing purposes. For example, you may create an icon using the square, line, and circle tool and want to work with them as a single item.

How to group objects together:

1. Use selector to select all objects you desire to group. Hold the shift key down to select more than one object at a time.
2. Choose **Group** from the **Graphics** menu.
3. Manipulate the group as necessary.

Ungroup

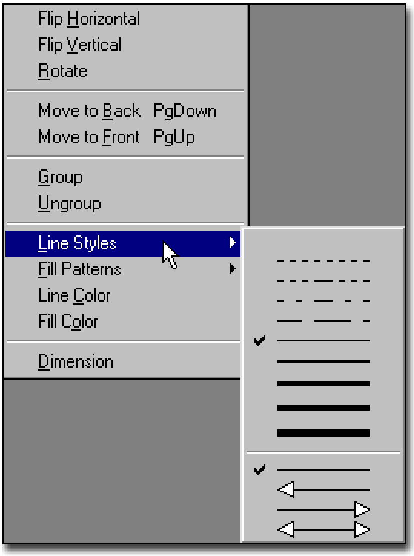
The Ungroup option allows you to ungroup several grouped objects.

How to ungroup previously grouped objects:

1. Use the selector to select the object you desire to ungroup.
2. Choose **Ungroup** from the **Graphics** menu.

Line Styles

You may choose different styles for the lines and borders of objects by choosing Line Styles in the Graphic Editor Graphics menu.



How to change the line style or border style of an object:

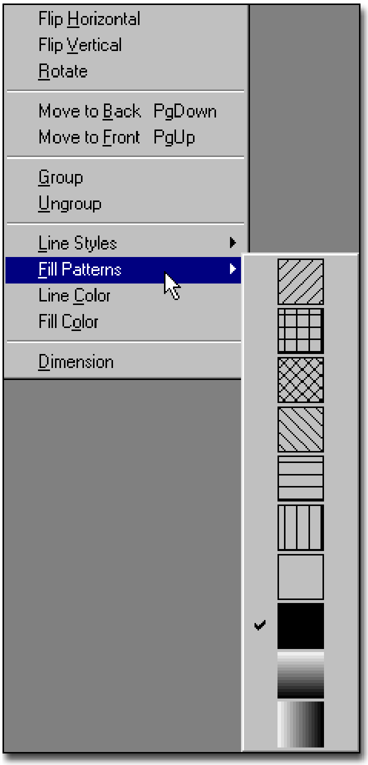
1. Select the desired object(s) using the Selector.
2. Choose **Line Styles** from the Graphic Editor **Graphics** menu.
3. Click on the desired style.

Please note

The arrowhead color is the same as the line color and is defined through the Line Color in the Graphics menu.

Fill Patterns

Various patterns may be used to fill each object by choosing Fill Patterns from the Graphic Editor Graphics menu.



How to change the fill pattern of an object:

1. Select the desired object(s) using the Selector.

2. Choose **Fill Patterns** from the Graphic Editor **Graphics** menu.

3. Click on the desired fill pattern.

Line and Fill Color

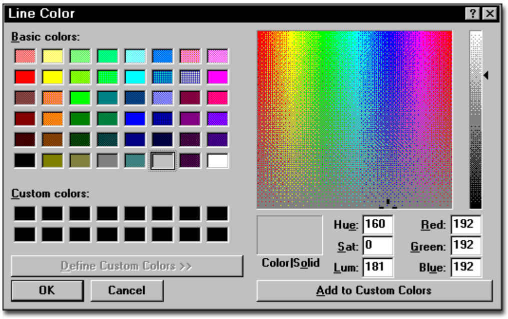
You may also select a custom color for your lines and graphics.

How to change line or fill color

1. Select the line or graphic you want to change.

2. Choose **Line Color** or **Fill Color** from the **Graphics** menu.

3. From the dialog below, click on the color you wish to use.

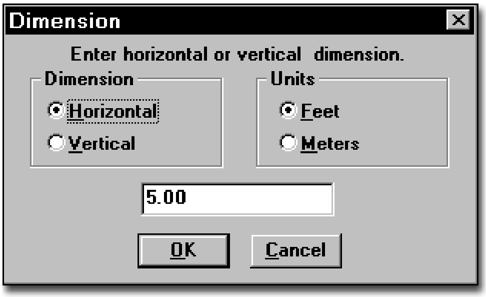


Hue: 160
Sat: 0
Lum: 181
Red: 192
Green: 192
Blue: 192

Dimension

You can define the height or width of an object in feet or meters from the Dimensions Dialog box.

This will determine how large the graphic appears when placed on a model layout. For example, to change the width of a graphic loaded in the Edit window to 5.00 feet, enter the following in the Dimensions dialog box:


Dimension

Enter horizontal or vertical dimension.

Dimension: ● Horizontal ○ Vertical

Units: ● Feet ○ Meters

5.00

How to change the graphic's dimensions:

1. Load a graphic into the Edit window.
2. Select **Dimension** from the **Graphics** menu.
3. Enter the new graphic dimension and click **OK**.

For example, a graphic of a computer is displayed in the Graphic Edit window. The size of the graphic in the Graphic Edit window is 1.40 x 1.00 as shown below.

Dimension: 1.40 x 1.00

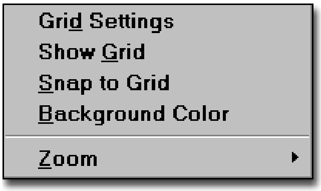
However, the computer is really 2 feet wide. To change the graphic's dimensions, select Dimension from the Options menu. Type 2 for the horizontal dimension and click OK.

Dimension: 2.00 x 1.43

Notice the vertical dimension is automatically adjusted to 1.43. You can only define either the horizontal or vertical dimension of the graphic. If you define one dimension, the software will automatically calculate the other according to the proportions of the graphic in the Edit window. If you know the graphic is actually 2.00 x 1.50 feet, you will need to adjust the proportions of the graphic accordingly by using various tools from the Tools button bar.

Options Menu

The Options menu controls the editing environment. With this menu you can use a grid to help align objects, edit that grid, and require objects to snap to it. Finally, the Options menu allows you to zoom in and out on the graphic so you can edit the graphic at different sizes.



Grid Settings Brings up the Grid dialog box for choosing the size, color and visibility of the grid.

Show Grid Causes the grid to appear in the background for editing purposes. If the grid is on, choosing this option will turn it off. This is the same as choosing Grid On in the Grid dialog box.

Snap to Grid Positions any object subsequently drawn or moved on the layout on the nearest grid line. This option works whether the grid is visible or not.

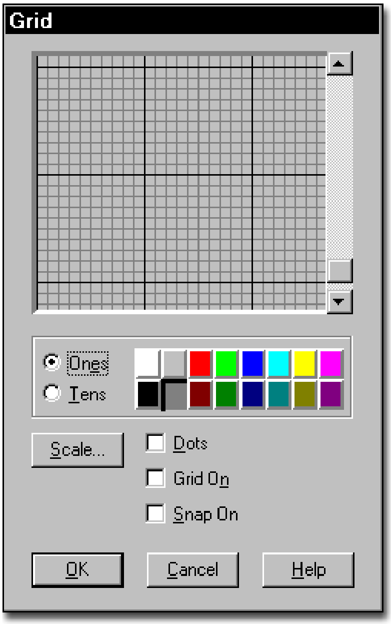
Background Color Brings up the color dialog box for choosing a background color for the layout window.

Zoom Shrinks or enlarges your view of the graphic by the percentage selected.

Grid Settings

The grid size is changed by using the scroll bar to the right of the grid dialog box. Move the scroll bar up to increase the grid size and move the scroll bar down to decrease the grid size. To change the color of the fine grid lines, select the Ones button and then choose a color. To change the color of the coarse grid lines, select the Tens button and choose a color.

Instead of viewing boxes as the grid units, you may choose dots by selecting the dots box. You may also choose to switch the grid, as well as the grid snap, on in this same area.



How to change the distance per grid unit in the Graphic Editor:

1. Select **Grid Settings** from the **Options** menu.

2. Click on the **Scale** button.

3. Define the distance per grid unit in either feet or meters and click **OK**.

Show Grid

How to show the grid on the layout window:

- Choose **Show Grid** from the **Options** menu.

Snap to Grid

How to have all new and edited objects snap to the grid:

- Choose **Snap to Grid** from the **Options** menu.

Background Color

How to change the background color of the edit window:

- Choose **Background Color** from the **Options** menu.

Zoom

How to magnify a graphic:

1. Choose **Zoom** from the **Options** menu.

2. Choose the level of magnification from the submenu.

Window Menu

The Window menu is used to manipulate the various windows in the Graphic Editor and follows the Microsoft Windows standard for Window menus.

Tile Causes all open windows to fit in the available screen space. Windows that may be hidden behind other windows will become visible. This is useful when desiring to view more than one graphic library on the screen at a time.

Cascade Causes all open windows to overlap such that the title bar of each window is visible.

Arrange Icons Takes all active iconized windows and arranges them neatly along the bottom of the screen.

Open Libraries Displays all graphic libraries currently open in the Graphic Editor. To switch to another open library, click on the desired name.

Library Edit Buttons

The Graphic Editor contains four library edit buttons: Edit, Save, Delete, and Clear.

Edit Retrieves a selected graphic from the library to the edit window.

Save Saves a graphic from the Edit window to the library. For a new graphic to be added to the library, the blank box at the end of the library must be selected. Otherwise, the graphic in the Edit window will replace whichever graphic's icon is selected.

Delete Deletes the selected graphic from the library.

Clear Clears the contents of the edit window.

How to edit a graphic:

- Double-click on the graphic in the library. This method clears the contents of the Edit window before loading the new graphic.

or...

- Select the graphic by clicking once on the graphic and then click the **Edit** button. This method clears the contents of the Edit window before loading the new graphic.

Manipulating Graphics

Among other things, a graphic can be reduced, enlarged, combined, and reordered.

How to change a graphic's size:

1. Select the graphic in the Edit window using the selector.

2. Choose **Group** from the **Graphics** menu.

3. Select one of the small gray boxes at the graphic edge to reduce or enlarge the graphic size. The selector will change to a cross-hair when you are near a gray box.

4. Drag the box to the desired size.

Several graphics can be combined together to create a single graphic. For example, a desk and a chair are separate graphics but would be easier to use if they were combined to form a single graphic.

How to combine two graphics:

1. Load the first graphic into the edit window.

2. Drag the second graphic into the edit window.

3. Position and size the two graphics as desired.

4. Click the blank box in the graphic library.

5. Click the **Save** button.

How to change the order of graphics in a graphic library:

- Drag a graphic to the desired location in the library.

Create New Graphics and Libraries

How to create a new graphic

1. Select **Graphic Editor** from the **Tools** menu.

2. Use the drawing tools to create the new graphic.

3. Click the **Save** button on the graphic editor.

4. The image appears at the end of the existing icons.

How to create a new graphic library

1. Select **Graphic Editor** from the **Tools** menu.

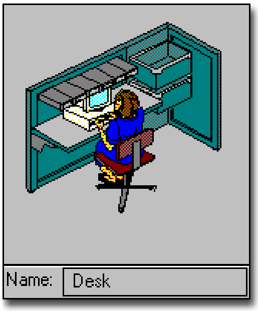
2. Select **New** from the **File** menu.

3. Use the drawing tools to create any graphics you wish to use in your library. You may also copy and paste graphics from existing libraries.

4. After you finish preparing each new image, click the **Save** button on the graphic editor.

Naming a Graphic

Graphics can be named for resources, locations, and entities, or merely for easier identification. When a named graphic is chosen while building a model, instead of entering a default name, such as Loc1, ProModel will enter the graphic's name, such as Desk. If the name already exists for a location or resource graphic, a number will be appended (Desk1, Desk2, etc.). If the name already exists for an entity graphic, a letter will be appended (ChairA, ChairB, etc.). This will make the model easier to understand and use.



How to name a graphic:

1. Enter the desired name at the bottom left of the screen where it says "Name."

2. Click the **Save** button to save the named graphic.

Please note

Correct syntax for location, resource, and entity names should be used when entering a name in the Name field if the graphic is intended to represent locations, resources, or entities. Graphic names are allowed to have spaces which automatically convert to underscores "_" when used for locations, resources, or entities.

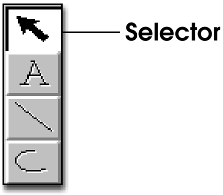
Graphic Tools Button Bar

The Graphic Tools Button Bar contains the tools necessary to create and edit a graphic's component objects. The drawing tools are the main tools through which graphics are created and edited in the Editing window. The drawing tools include the following:

- Selector
- Entity Spots
- Text
- Status Lights
- Lines
- Flip Horizontal
- Arcs
- Flip Vertical
- Triangles
- Rotate
- Regular Squares and Rectangles
- Cut
- Rounded Squares and Rectangles
- Copy
- Circles and Ellipses
- Paste
- Chords
- Step Back
- Pies
- Step Front
- Polygons
- Line Color
- Raised Squares and Rectangles
- Fill Color

Selector

The Selector is a pointing device that allows you to select one or more objects of a graphic. It also allows you to move, size, and shape all graphic objects.



How to move an object:

1. Click on the Selector button.

2. Drag the object to the desired location.

How to size or shape an object:

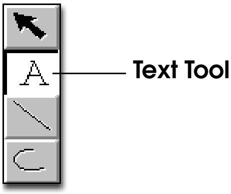
1. Click on the Selector button from the button bar.

2. Select the desired object.

3. Drag the sizing points. The arrow will turn into a cross-hair when it approaches the sizing points.

Text Tool

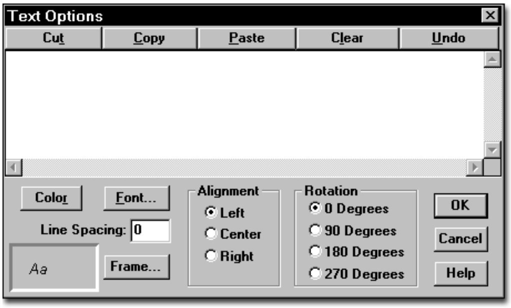
Text may be placed anywhere in the graphic by using the Text tool.



How to place text in a graphic:

1. Select the Text tool from the button bar.

2. Click where the text is to appear. The Text dialog box will open.



Line Spacing: 0

Alignment
● Left
○ Center
○ Right

Rotation
● 0 Degrees
○ 90 Degrees
○ 180 Degrees
○ 270 Degrees

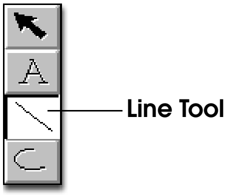
3. Enter the desired text in the Text dialog box below. Set the desired options for the text. When finished, click the **OK** button.

How to edit text already in the graphic:

- Double click on the text.

Lines

Lines may consist of several segments and are drawn using the line tool.



How to draw lines in the graphic editor:

1. Select the Line tool.

2. Click the left mouse button where the line is to begin.

3. Move the mouse to the end of the line segment.

4. Click the left mouse button to create a joint and begin the next segment.

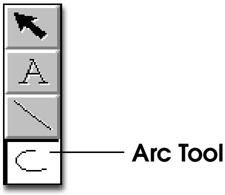
5. Double click the left mouse button or click the right mouse button to end the line.

Please note

Lines can be drawn at 15 degree increments by holding the shift key while moving the end of a line segment.

Arcs

Arcs are drawn using the Arc tool.



How to draw an arc in the graphic editor:

1. Select the Arc tool from the Tools window.

2. Press the left mouse button at one end of the desired arc.

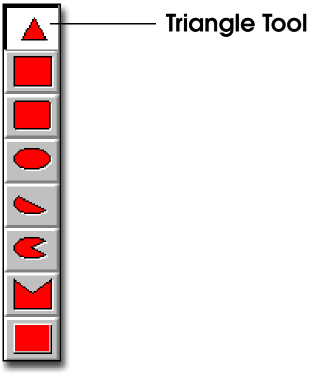
3. Drag the mouse to the other end of the arc and release the left mouse button.

Please note

To have the arc bow left, start the arc from the top and drag down. To have the arc bow right, start the arc from the bottom and drag up.

Triangles

Triangles are drawn using the Triangle tool.



How to draw a triangle in the graphic editor:

1. Select the Triangle tool from the Tools button bar.

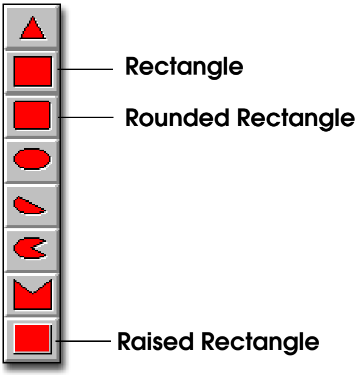
2. Press the left mouse button where the center of the base of the triangle is to be located.

3. Drag the mouse until the triangle's base is the desired size, but do not release the mouse button.

4. Press the shift key and move the mouse to adjust the size of the other two sides. Release the mouse button when done.

Squares and Rectangles

Squares and rectangles may be drawn using the regular rectangle tool, the rounded rectangle tool, or the raised rectangle tool.



How to draw a rectangle in the graphic editor:

1. Select either the regular, rounded, or raised rectangle tool from the Tools window.

2. Press the left mouse button at the top left corner of the desired rectangle.

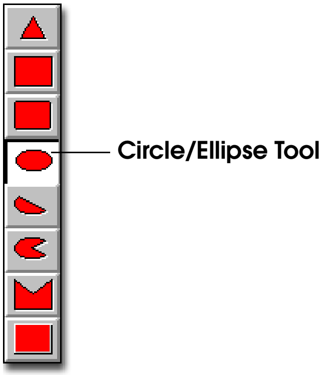
3. Drag the mouse to the lower right corner of the rectangle and release the mouse button.

How to draw a square in the graphic editor:

- Hold the shift key while drawing or editing a rectangle.

Circles and Ellipses

Circles and ellipses may be drawn using the ellipse tool.



How to draw an ellipse in the graphic editor:

1. Select the ellipse tool from the Tools window.

2. Press the left mouse button at one end of the desired ellipse.

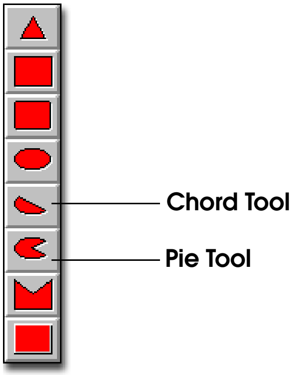
3. Drag the mouse to the other end of the ellipse and release.

How to draw a circle in the graphic editor:

- Hold the shift key while drawing or editing an ellipse.

Chords and Pies

Chords and pies are drawn using the Chord tool and Pie tool.



How to draw a chord or pie in the graphic editor:

1. Select the chord tool or pie tool from the tools menu.

2. Press the left mouse button at one end of the desired chord or pie.

3. Drag the mouse to the other end of the chord or pie and release the left mouse button.

How to draw a circular chord or pie in the graphic editor:

- Hold the shift key while drawing a chord or pie.

How to adjust the size of the "slice" in the chord or pie after the graphic has been drawn:

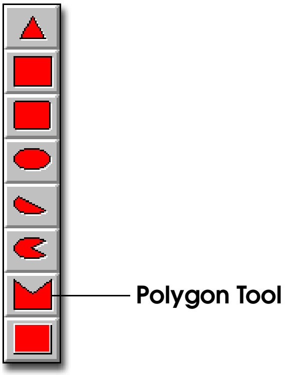
1. Select a corner of the "slice" on the graphic.

2. Press the left mouse button at the corner of the "slice."

3. Drag the mouse to the desired position of the "slice" and release the left mouse button.

Polygons

Polygons are drawn using the Polygon tool.



How to draw a polygon in the graphic editor:

1. Select the Polygon tool from the Tools window.

2. Click the left mouse button to begin the first point of the polygon.

3. Click the left mouse button at each successive point of the polygon.

4. Click the right mouse button or double-click the left mouse button to end the polygon.

Please note

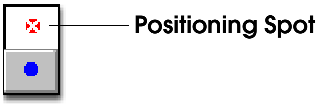
The sides of a polygon may be drawn at 15 degree increments by holding down the shift key while moving the mouse to the next vertex.

Positioning Spot

A positioning spot controls the positioning of an entity on a location, resource, or path. It also controls the positioning of an entity or resource on a path. There are two types of positioning spots: entity spots and alignment spots.

Entity spots can be defined for a graphic in either the Graphic Editor or in the Locations module. Alignment spots can only be defined for a graphic in the Graphic Editor. In the Graphic Editor, the button showing the red circle with the white X represents the positioning spot. The default type is entity spot. To change the type, the user may double click on the spot to display the Spot Type dialog. This allows the user to define the type of positioning spot.

A graphic may have any number of entity spots. A graphic may also have any number of alignment spots. However, only the first alignment spot defined will be used. An entity or alignment spot is ignored if it is inapplicable for the model element it is used to represent. The following definitions explain uses of the entity and alignment spots for locations, entities, and resources.



How to place a positioning spot on an icon:

1. Select the positioning spot tool from the button bar.
2. Click on the icon where the entity is to appear.

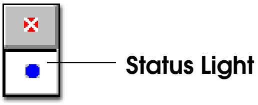
Location Graphics May only use entity spots. Whenever an entity arrives at the location, the entity graphic will be placed on the first entity spot defined for the location. The next arriving entity will use the entity spot defined second, and so forth. If entity spots are defined for a location graphic in the Locations module, they are used ahead of any entity spots defined in the Graphic Editor. If no entity spot is defined for the location graphic, no entity is shown on the location.

Entity Graphics Use only alignment spots. When an alignment spot is defined for an entity graphic, the entity graphic will be positioned so the alignment spot of the entity graphic and the entity spot for the location or resource graphic are aligned. If the entity is traveling along a path, the entity graphic will move along the path with the alignment spot and the path segment or node aligned. If no alignment spot is placed on an entity graphic, the center of the entity graphic is used for alignment.

Resource Graphics May use both entity spots and alignment spots. An entity spot on a resource graphic may be used to locate an entity a resource is carrying. An alignment spot can be placed on a resource graphic so that when the resource travels along a path, the resource graphic will move along the path with the alignment spot and the path segment or node aligned. If no alignment spot is placed on a resource graphic, the center of the graphic is used for alignment.

Status Lights

A status light is a circle that changes color depending on the status of a location. A status light can be placed anywhere relative to a location to show the status or current state of the location. At run time, a window can be displayed showing what status each color represents.



How to place a status light on an icon:

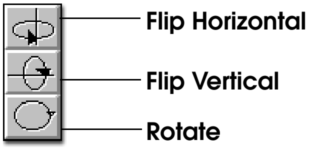
1. Select the Status Light tool from the button bar.
2. Click on the icon where the status light is to appear.

Please note

Status lights for location graphics may also be defined in the Locations Editor.

Flip and Rotate

Objects may be flipped about the horizontal axis and vertical axis or rotated by 90 degrees using the flip and rotation tools from the Tools button bar.



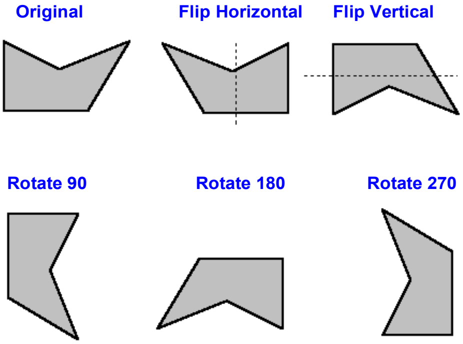
How to flip or rotate an object:

1. Select the desired object(s).

2. Click on the Flip Horizontal, Flip Vertical, or Rotate buttons.

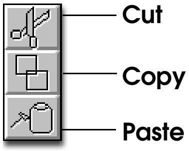
Using the flip and rotate buttons is the same as choosing flip and rotate options under the Graphics menu.

The figure below shows an object that has been flipped horizontally, vertically, and rotated through all phases.



Cut, Copy, and Paste

To speed the development of complex graphics, you may cut, copy, and paste objects from one area of the workspace to another. Each of these buttons works exactly the same as the corresponding item from the Edit menu.



How to cut an object:

1. Select the desired object(s) using the Selector.

2. Click the Cut button. The object is removed from the Edit window but remains in the Graphic Editor’s internal clipboard.

How to copy an object:

1. Select the desired object(s) using the Selector.

2. Click the Copy button. The original object remains on the screen and a copy is placed on the Graphic Editor’s internal clipboard.

How to paste an object:

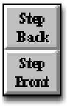
1. Click on the Paste button. The contents of the internal clipboard are pasted next to the last object cut or copied.

2. Move the new object to the desired location using the Selector.

Step Back and Step Front

You can move an object behind or in front of another object. The Step Back option allows you to move a selected object behind another object. The Step Front option allows you to move a selected object in front of another object.

For example, suppose you have five graphic objects displayed and you want to move the top object to the third object. You can use the Step Back or Step Front option.



How to move an object behind or in front of another object:

1. Select the object to move using the Selector.

2. Click on the Step Back or Step Front tool from the button bar.

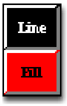
3. Continue to press the Step Back or Step Front button until the selected object is behind the desired object.

Please note

If you would like to move an object behind or in front of all objects, use the Move to Back or Move to Front option in the Graphics Menu. Alternatively, use the Page Up or Page Down keys.

Line and Fill Color

An object's line and fill colors can be chosen using the Line tool and the Fill tool. You may use one of the predefined colors or create your own custom color. Each tool's color changes according to the color chosen.



How to define the line color or fill color:

1. Select the object(s) to change using the Selector.

2. Click on the Line or Fill tool from the button bar.

3. Choose the desired color. The tool will change to the color specified.

Please note

The line color and fill color can also be defined in the Graphic Editor Options menu.

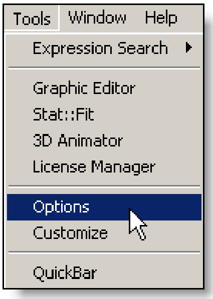
Editing a Library Graphic

Various editing functions allow you to alter the objects that comprise a library graphic. These functions may be applied to the entire graphic or to one of the objects from which the graphic is constructed. The following is a description of how to edit a graphic (All mouse actions are performed using the left button unless stated otherwise).

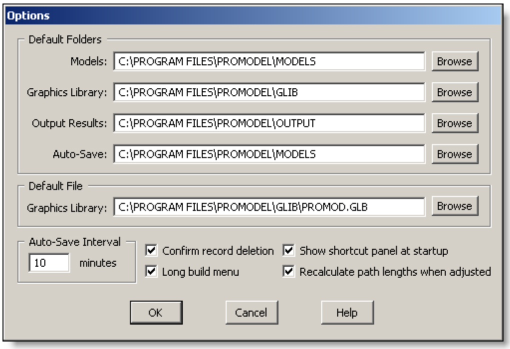
TO...	DO THIS
Select an object.	Choose the Selector tool and click on the object.
Select multiple objects.	Drag in an empty region until a bounding rectangle encompasses the objects. or Shift+Click on each of the objects you want selected (Shift + Click again on a selected object deselects it).
Move one or more selected objects.	Drag the selected object(s).
Delete selected objects.	Press the Delete key. or Select Cut from the Edit menu (This method puts the object on the clipboard for subsequent pasting).
Copy selected objects.	Press the Copy button to copy the selected objects to the clipboard. Then press the Paste button to place a copy of the selected objects into the Edit window. or Choose Copy from the Edit menu. Then choose Paste from the Edit menu.
Edit text.	Double click on the text to bring up the text editor dialog box.
Change the shape of a selected object.	Drag one of the sizing points of the selected object.
Add a vertex to a selected line or polygon.	Right click on the line or polygon where the vertex is to be added.
Delete a vertex of a selected line or polygon.	Right click on the vertex.
Change the fill pattern for a selected object.	Choose Fill Patterns from the Graphics menu and select the desired pattern.
Change the color of a selected object.	Click on the Line or Fill Color button with one or more objects selected.
Change the line style for a selected object.	Choose Line Styles from the Graphics menu. Then choose the desired line style.
Flip or rotate a selected object.	Click on the flip or rotate button with one or more objects selected.
Move a selected object in front of another object.	Click on the Step Front button until the selected object is in front of the other objects.
Move the selected object in front of all other objects.	Choose Move To Front from the Graphic menu. or Press the Page Up key.
Move a selected object behind another object.	Click on the Step Back button until the selected object is behind the other objects.
Move a selected object behind all other objects.	Choose Move To Back from the Graphic menu. or Press the Page Down key.
Nudge a selected object one pixel left.	Press the Left arrow key.

TO...	DO THIS
Nudge a selected object one pixel right.	Press the Right arrow key.
Nudge a selected object one pixel up.	Press the Up arrow key.
Nudge a selected object one pixel down.	Press the Down arrow key.
Size a background graphic proportionally.	Select the graphic, group it, then size using the handles.
Create a perfect circle or square.	Select a graphic and, while holding down the shift key, size the graphic.

Options



The Options dialog contains default folders, selections for displaying the long build menu, defaults for record deletion, and the time between auto-saves.


Options

Default Folders

Models: C:\PROGRAM FILES\PROMODEL\MODELS

Graphics Library: C:\PROGRAM FILES\PROMODEL\GLIB

Output Results: C:\PROGRAM FILES\PROMODEL\OUTPUT

Auto-Save: C:\PROGRAM FILES\PROMODEL\MODELS

Default File

Graphics Library: C:\PROGRAM FILES\PROMODEL\GLIB\PROMOD.GLB

Auto-Save Interval: 10 minutes

☑ Confirm record deletion

☑ Long build menu

☑ Show shortcut panel at startup

☑ Recalculate path lengths when adjusted

Default Folders These fields contain the default folders for your model.

Default File: Graphics Library This field allows you to specify a default graphic library file for every new model you create.

Auto-Save time interval Allows you to select how often ProModel will automatically save your model.

Confirm record deletion Use this option to have ProModel display a dialog box confirming that the user wants to delete a record from an edit table.

Show shortcut panel at start-up Check this option to display the shortcut panel at start-up.

Long build menu Allows the user to view the long build menu.

Recalculate path lengths when adjusted Recalculates the time or distance of a path network or conveyor as it is graphically lengthened or shortened.

Directories

The Directories section of the Settings dialog allows you to specify which drives and folders to use for storing models, graphic libraries, and output results.

How to change the default folders:

1. Select **Options** from the **Tools** menu.
2. Click on the Browse button next to the Folder name you would like to change.
3. Specify the desired folders for models, graphics libraries, output results, and auto-save.
4. Click **OK**.

Long Build Menu

The Long Build Menu option reorganizes the Build menu. The long build menu takes the first section of the More Elements submenu and places it in the Build menu. This includes attributes, variables, arrays, macros, and subroutines. Using the long build menu is especially helpful when using these elements frequently.

How to display the long menu:

1. Choose **Options** from the **Tools** menu.
2. Check the **Long build menu** option.

Please note

To display the short menu, follow the same procedure above and uncheck the Long build menu option.

AutoSaving Files

ProModel automatically saves the open model every few minutes, which is useful in the event of unforeseen crashes and power outages. ProModel uses a model file called “AUTOSAVE.MOD” for all autosaves and only modifies the original file when Save is chosen from the File menu.

How to specify the amount of time between AutoSaves:

1. From the **Tools** menu, select **Options**.
2. In the **Time between autosaves** field, enter the time.
3. Click **OK**.

Please note

Models are always autosaved at the start of a simulation run. To deactivate the auto-save feature, set the time between auto-saves to 0.

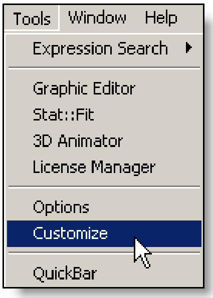
How to specify the autosave directory:

1. From the **Tools** menu, select **Options**.
2. In the **Auto-save** field, enter the directory path you wish to use.
3. Click **OK**.

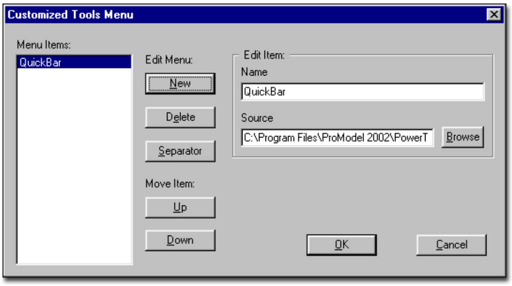
Customize

You can add direct links to applications and files right on your ProModel toolbar. Create a link to open a spreadsheet, a text document, or your favorite calculator–it's your menu.

To create or modify you Custom Tools menu, select Customize from the Tools menu.



This will pull up the Custom Tools dialog window. The Custom Tools dialog window allows you to add, delete, edit, or rearrange the menu items that appear on the Tools drop down menu in ProModel.



Name: QuickBar

Source: C:\Program Files\ProModel 2002\PowerT

Three tools come already added to the Custom Tools menu by default: QuickBar, Models to Go, and Model Package Association. These tools can be added to or deleted from the Tools menu.

Add a new menu item:

1. Click on the New button.

2. In the Name text box, type the name of the item you will be referencing, as you want it to appear on the drop menu.

3. Click the Browse button.

4. Use the browse window to find the file or application that you wish to have launched, then double click on it or click Open.

5. The file or application name and path will appear in the Source window.

Delete a menu item

1. In the Menu Items list, click on the name of the item you would like to delete.

2. Click the delete button.

Edit a menu item

1. In the menu Items list, click on the name of the item you would like to edit. Its properties will appear in the Edit Item panel, on the right.

2. Change the name that appears in the Name text box.

3. To redirect the item to another file or application, change the path and file name in the Source text box by either typing in a new source, or using the Browse window.

Rearrange items on the menu

1. You can rearrange the order in which menu items appear on the Tools menu, or add separators to divide them into logical groups. To move an item up in the list, select the item you wish to move.

2. Click on the Up or Down button to move the menu item’s position in the menu list.

Add a Separator

1. Clicking on the Separator button adds a Separator line to the drop down list. In the Menu Items list, this line is represented as --<Separator>--, in the Tools menu it appears as a single solid line.

2. The Separator line is inserted into the menu list above the highlighted Menu Item. Its placement can then be adjusted similar to other menu items, using the Up and Down buttons

Please note

Custom Tool settings are saved in your Windows Registry. They are not a part of your model file or model package. If you have multiple installations of ProModel Corporation products (such as a LAN version, MedModel, or ServiceModel) your Customized Tool menu will be available to your other ProModel Corporation products as well.

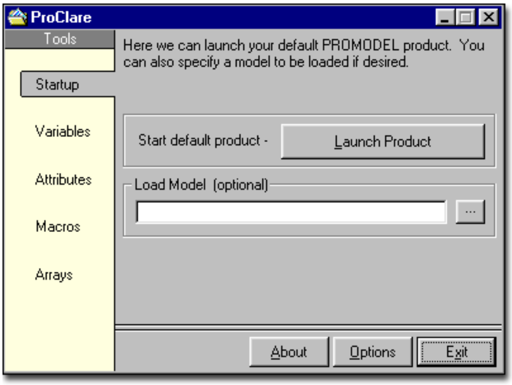
Power Tools

Packaged with ProModel are several powerful tools that can help you better use and understand your ProModel software and the statistical data it creates. These tools include:

- ProClare
- ProSetter
- Shift Library
- ProActiveX

Any of these tools can be added to the custom Tools menu using the steps outlined in the section “Add a new menu item:” on page 336.

ProClare



Here we can launch your default PROMODEL product. You can also specify a model to be loaded if desired.

ProClare is a quick way to add new variables, attributes, macros and arrays into your ProModel Corporation product without having to leave where you are in the model building process. This is extremely useful if, for example, you are into heavy logic and need to add a new variable. With this tool you can add the new variable without losing your place.

Tools

Startup Loads the default ProModel Corporation product for model building and editing. You can choose to specify a model to be loaded.

Please note

When using the ProClare tools, if no ProModel Corporation products is currently open, ProClare will launch the default ProModel Corporation product for you.

Variables This will allow the addition of new variables into the model. The name, initial value, type, and RTI stats can be set here.

Attributes This allows for the addition of new attributes. The attribute name, type, and value can also be configured.

Arrays This allows the user to add arrays, with their corresponding name, dimensions and type.

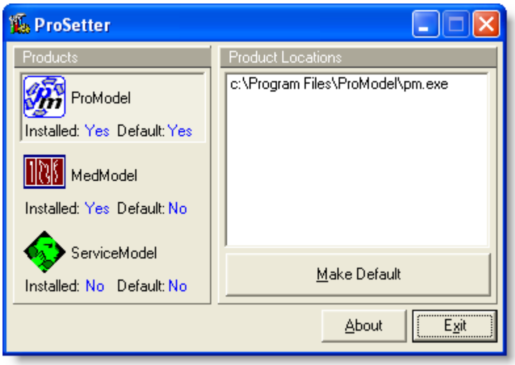
Options

Select the Options button to customize ProClare in these additional ways:

Always on Top This option allows ProClare to always be on top of all other windows, which is useful if you plan on adding a lot of new items.

Auto Roll-up When enabled, ProClare is reduced to just the title bar when the mouse cursor is outside the window. When the cursor is inside the window, the window is restored to normal size.

ProSetter



ProModel
Installed: Yes Default: Yes

MedModel
Installed: Yes Default: No

ServiceModel
Installed: No Default: No

c:\Program Files\ProModel\pm.exe

This tool is useful only if you have more than one ProModel Corporation product installed. This includes having more than one installation of ProModel, or having every ProModel Corporation product installed.

When you start this tool, it will scan all your hard-drives for ProModel Corporation products (ServiceModel, ProModel, MedModel, or multiple installations of any one of these). To change the default product, click which product you want to be your default. If you have multiple copies of a product, select which location. Having made your selection, you must then click the "Make Default" button to process the change. When you make a product the default product, it will modify the registry, indicating to Windows that this product is now your default product. It will also reset your icon cache to reflect the new file association.

Products

This tool supports ServiceModel, ProModel and MedModel. It supports multiple installations of these products as well.

Each product has 2 labels: Installed and Default. Both labels say either Yes or No next to them to reflect whether it is installed, and indicate if it is the default product.

Locations

When you click on a product, it will populate the location box with all of the locations to which that product is installed. The default product location will have a "(Default)" caption next to it to indicate that it is the default.

Shift Library

For your convenience, predefined shift files are now included with your ProModel Corporation product. When installed they will appear in a Shift Library folder within your product installation folder i.e. C:\ProModel\Shifts\shift_file.sft.

These files have common nomenclature to simplify the process of identification and implementation of the proper shifts. Each shift file is named in the following fashion: ####x_####x_#(_###).sft. The first four numbers tell you what time the shift starts along with a letter to designate am or pm. The second four numbers specify what time the shift ends along with a letter designating am or pm. The third number specifies how many days of the week for which there is a shift definition. The numeral 1 means the shift occurs one day out of seven; seven means the shift is defined for every day of the week; a Monday through Friday schedule would be represented with the number five. The last value in brackets is only used when breaks are defined for the shift. The first of these three numbers indicates the number of breaks in a day, and the next two numbers indicate the cumulative minutes of those breaks.

Example

7a_330p_5_360.sft

This file name represents a shift that starts at 7:00 a.m. and ends at 3:30 p.m., five days a week. There are three breaks defined, one for 15 minutes, one for 30 minutes, and another for 15 minutes. These breaks add up to a cumulative 60 minutes of break time, represented by 360.

ProActiveX

Overview

The ProActiveX spreadsheet is designed to help you understand how to use ProModel's ActiveX components. The worksheets and macros give clear and functional examples of how to access all currently enabled data elements.

The ProActiveX spreadsheet may be used "as is" or modified to suit your specific needs. With ProActiveX and a fundamental knowledge of Visual Basic for Applications, you can create customized user interfaces for ProModel–or let our consulting division do it for you!

ProActiveX.xls has nineteen data worksheets corresponding to the module with the same name in ProModel. Additionally, there is one hidden sheet with data for drop-down lists used on the other sheets.

Please note

ProActiveX is only compatible with Excel 2000 or newer.

Worksheets

Drop-down lists, check boxes and text boxes

Many of the worksheets in ProActiveX have drop-down lists which make it easier to fill in the required data. In all cases, the value you select will be inserted into the selected cell or range of cells. However, the assigned macros do not prevent you from selecting a different column than the one with the drop-down used.

Using a combo box with a pre-defined list is one way to ensure consistency in the data. In most cases, the numeric value of the list selection is temporarily recorded on the hidden sheet (named **Hidden Sheet**). On the **Simulation Options** sheet, instead of filling in a cell or range, the value shows in the drop-down box itself and the numeric value is hidden in the cell behind it.

The **Simulation Options** sheet also contains several check boxes and a text box. The values for the check boxes are hidden in the cells behind them. The method used for hiding the values is simply to make the text color the same as the background color.

Not Enabled and Partially Enabled

A few of the worksheets have data columns defined for methods which have not yet been implemented in the ProModel **Data Object.** These columns have been included to acknowledge that these elements were not forgotten when ProActiveX was created–these data elements will be enabled in future releases of ProModel.

Shift Editor Button

The Shift Assignments worksheet has a button to the far right labeled **Shift Editor**. Clicking on this button will show the Shift Editor, just like when you invoke the menu call from your default ProModel Corporation product.

Browse Button

The General Info worksheet has a **Browse...** button that uses the Microsoft Common Dialog to display a file selection window.

Comments

Several worksheets have comments in the cells of the title rows. These comments are included to give you important information about the data contained in that column–like limitations or formatting requirements.

Panes and Sections

In each worksheet where the data columns go beyond the right-hand border of a single screen, separate panes have been created and ‘frozen’ to allow you to scroll through the columns without moving the column that identifies the record. The title rows have also been frozen, so that the user can scroll down the sheet without losing sight of the titles.

In each worksheet, title labels spanning more than one column indicate that all the columns beneath that title pertain to that type of information. For example, on the Resources worksheet, there are several columns for downtime data which have duplicate titles, but different section labels. In some cases, the labels are used merely to make the titles easier to read.

Controls Sheet

The first sheet of the ProActiveX file is the Controls Sheet. The options available on this sheet are described below.

Get (All)

The **Get** button copies ActiveX enabled data elements from the currently open model to the ProActiveX spreadsheet.

ProModel must be running and a model loaded. When all of the data has been copied to the spreadsheet, a message box will appear saying, “Process Complete.”

Build Model

The Build Model button copies all ActiveX enabled data elements from the ProActiveX spreadsheet to the currently open model.

ProModel must be running and a model loaded. When all of the data has been copied to the model, a message box will appear saying, “Process Complete.”

Save Model

The **Save Model** button saves all updates to the loaded model. ProModel must already be open with a model loaded.

Run Simulation

The **Run Simulation** button starts the simulation for the loaded model. To view the simulation, you will have to manually select ProModel from your Windows Task Bar, to bring it forward.

Open ProModel

The **Open ProModel** button starts ProModel, but does not load a model.

Load Model

The **Load Model** button gives the user an opportunity to specify a model and then open it in ProModel. If ProModel is not already running, the **Load Model** button will start ProModel and then load the selected model. This button gives you an example of a custom dialog box.

Macros

All of the processing logic for ProActiveX is contained in Excel macros. In most cases each macro is in a separate module. The controlling modules are named **Get_ALL_Model_Data** and **Pop_ALL_Model_Data.** There are also macros in some of the Excel Sheet Objects. For example, Sheet04 (Resources) has subroutines that dynamically fill in the Path Network drop-down list (using the information on the Path Networks sheet) every time the Resources sheet is activated. This is one method for keeping the drop-down list in sync whenever you update the data.

When the user clicks on the **Build All** button on the Controls Sheet, the **create_model** subroutine is started (found in the POP__ALL_Model_Data module). This subroutine then calls subroutines in other modules to process each type of data. The order in which the subroutines are called is important, since many of the modules in ProModel depend on information in other modules. For example, the Locations module must be built before the Path Networks module, because the Path Networks module references Locations.

All of the subroutines in ProActiveX use the constants established in the **PM_CONSTANTS** module. You will also find this module recorded in the file called **PM_CONSTANTS.bas**. The main purpose of the constants is to provide a method for accessing the parameters to the ProModel properties and methods data, without having to remember the numbers associated with each table and field.

There are some other useful subroutines that are used throughout the program to do things like showing the right error message or replacing Excel style carriage returns with ProModel style carriage returns.

In many cases when you look at the tables in ProModel, what you actually see on the screen is a main table and one or more sub tables.

The subroutines in ProActiveX must process the data for all tables simultaneously by using **For...Next** or **Do While...** loops.

In many of the data tables, the first field is a **Name** or **ID** field that identifies each record in the table and does not allow duplicates. In the subroutines that populate these types of tables, the logic is designed to search for an existing ProModel record with the same name or ID. If a record is found, all of its fields will be updated. If no record is found, a new one will be appended. In tables that have no identifying field, records are always appended.

Promodel Player

Promodel Player is a browser plug-in for Microsoft's Internet Explorer that allows users to view the animation and statistics of a model. PMplayer is automatically installed on your computer as you install ProModel.

Viewing Simulation Models

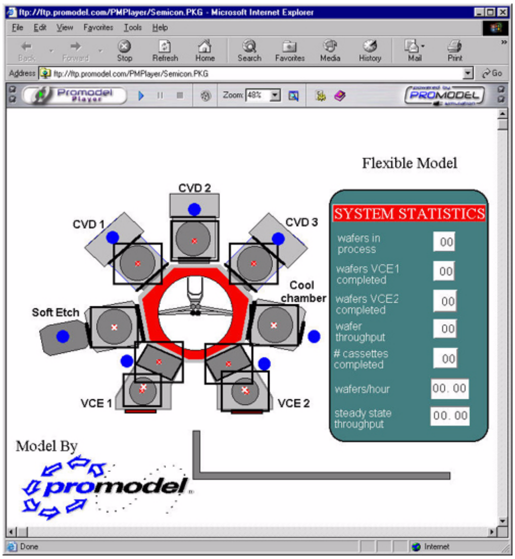
To view, or "play," a simulation model in Windows Internet Explorer, you may:

- Select the model file in Windows Explorer, then right-click on the package and select Play from the menu that appears. Three packages, Semiconductor.pkg, DaySurgery.pkg, and TollPlaza.pkg, are installed on your computer when you install ProModel. The are found in the following directory: C:\Program Files\PMPlayer.

  or,
- View a model via HTML links to the desired model package. Model links are available from the ProModel Corporation Web site at www.promodel.com. You cannot open Promodel Player independently of selecting a model package.

When you open a package on your computer, or click on a package link on the Internet, an Internet Explorer window will open and you will see the following image in your browser window while the package is loaded:

After the model has been loaded, the layout window of the model appears in the main browser window.


ftp://ftp.promodel.com/PMPlayer/Semicon.PKG - Microsoft Internet Explorer

Address: ftp://ftp.promodel.com/PMPlayer/Semicon.PKG

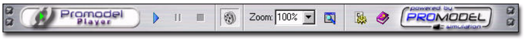
Zoom: 48%

SYSTEM STATISTICS

wafers in process	00
wafers VCE1 completed	00
wafers VCE2 completed	00
wafer throughput	00
# cassettes completed	00
wafers/hour	00. 00
steady state throughput	00. 00

Using Promodel Player

Simulations in Promodel Player are controlled through the Toolbar.


Zoom: 100%

▶ : Runs the animation.

‖ : Pauses the animation.

■ : Stops the animation.

🎞 : Toggles the Animation on/off. Turning the animation off increases the speed of the simulation.

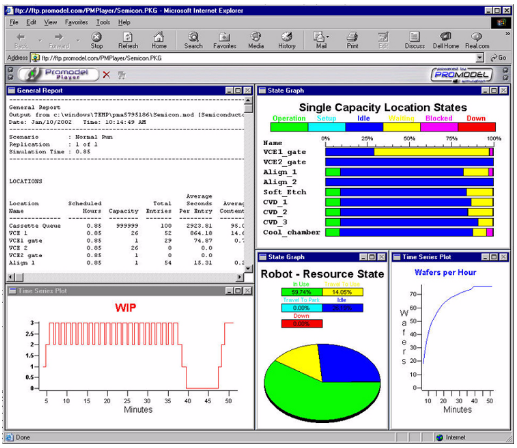
Zoom: 100% :Allows you to zoom in or out on the layout.

: Toggles the Auto Zoom on/off. When selected, this option will zoom the animation to fit within the browser window.

: Opens an options window, which gives you several options for controlling your animation.

: Opens this Help file.

To run the model animation, click Promodel Player's play button. If you have selected to view the output, as soon as the simulation is complete, the output statistics and graphs will open in the browser window. Only predefined output graphs will come up, the user cannot define new graphs through Promodel Player.



```
---------------------------------------------------------------------------
General Report
Output from c:\windows\TEMP\pma5795186\Semicon.mod [Semiconducto
Date: Jan/10/2002   Time: 10:14:49 AM
---------------------------------------------------------------------------
Scenario        : Normal Run
Replication     : 1 of 1
Simulation Time : 0.85
---------------------------------------------------------------------------


LOCATIONS


                                                          Average
Location         Scheduled                   Total        Seconds     Averag
Name                 Hours    Capacity     Entries      Per Entry    Content
-------------    ---------    --------     -------      ---------    -------
Cassette Queue        0.85      999999         100        2923.81       95.0
VCE 1                 0.85          26          52         864.18       14.6
VCE1 gate             0.85           1          29          74.87        0.7
VCE 2                 0.85          26           0            0.0
VCE2 gate             0.85           1           0            0.0
Align 1               0.85           1          54          15.31        0.2
```

If there are any bar graphs defined, as in the Semicon report, the user can get a detailed pie chart of the individual bar by left-clicking on the bar of the desired statistic.

To close the Output window and return to the animation window, click on the red "x" on the toolbar.

Please Note

A user who has installed this plug-in, but has not installed any other ProModel Corporation products will not have access to any of ProModel's model building capability, they may only view the simulation animation and the statistical output results. Users who have a valid ProModel Corporation registration key will have access to Promodel Player Gold, which is described in the next section.

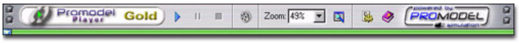
Promodel Player Gold

If you have a valid registration key on your computer for a ProModel Corporation product, Promodel Player will automatically launch Promodel Player Gold.

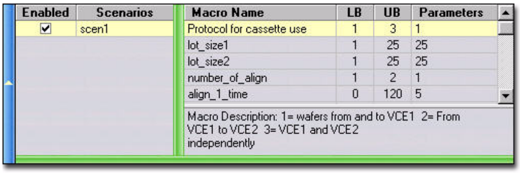
The Gold version of Promodel Player is functionally similar to the regular version of Promodel Player, but it contains the added functionality of scenarios and macros.

Promodel Player Gold allows you to view values for scenarios that are packaged with any .pkg file.

The scenario viewing area is accessible through the green bar underneath the Promodel Player toolbar.



Clicking on the arrow on the far left of the green bar will open the scenario and macro window.



Enabled	Scenarios
☑	scen1

Macro Name	LB	UB	Parameters
Protocol for cassette use	1	3	1
lot_size1	1	25	25
lot_size2	1	25	25
number_of_align	1	2	1
align_1_time	0	120	5

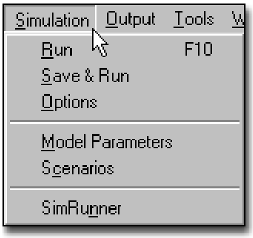
Macro Description: 1= wafers from and to VCE1 2= From VCE1 to VCE2 3= VCE1 and VCE2 independently

From this window you can select the scenarios you wish to run, and view the available macros.

Chapter 9: Running the Model

Simulation Menu

All of the run-time controls are accessed through the Simulation menu located on the menu bar. This menu contains options for running a model, specifying multiple replication statistics, defining scenario data, and other extended run-time options.



Each of the selections available on the Simulation menu is explained below.

Run Choose Run to begin simulating the current model using the options previously selected in the Simulation Options dialog box. Choosing Run does not save the models to the current model file. However, the model data will be saved to the file AUTOSAVE.MOD. To run several scenarios, you must select Run Scenarios from the Scenarios dialog mentioned below. Otherwise, it will run the RTI default values.

Save & Run Saves the current model, then runs it.

Options Select Options to bring up the Simulation Options dialog box used to specify important run-time information such as run length, warm-up period, number of replications, and multiple replication statistics. (See “Simulation Options” on page 348.)

Model Parameters Choose Model Parameters to open the Model Parameters dialog box. The dialog box allows you to modify the current settings for the RTI (Run-Time Interface) parameters defined in the Macros module. (See “Run-Time Interface” on page 242.)

Scenarios Select Scenarios to open the Scenarios dialog box. The dialog box allows you to define different scenarios using defined RTI (Run-Time Interface) parameters. (See “Scenarios” on page 353.)

SimRunner SimRunner takes your existing ProModel simulation models, evaluates them *for you*, then performs tests to find better ways to achieve the results you desire. (See “SimRunner” on page 370.)

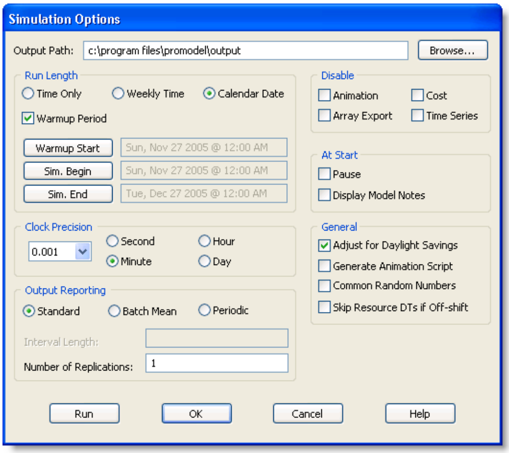
Simulation Options

The Simulations Options dialog provides you with a number of options to control the simulation, such as run length, warm-up time, clock precision, and the output. You also control the type of statistics reporting you want from the simulation, including period length and number of replications.

In addition to reporting standard statistics based on one or more replications of a simulation, ProModel allows you to average statistics across intervals of a single replication (batch mean) for analyzing steady-state systems or take the average of the average statistics for specific time periods over multiple replications (periodic) for non-steady state systems. You may choose from three statistical reporting options.

How to open the simulation options dialog:

1. Select the **Simulation** option from the menu bar.
2. Select **Options** to open the Simulation Options dialog.



Simulation Options

Output Path: c:\program files\promodel\output [Browse...]

Run Length
○ Time Only ○ Weekly Time ● Calendar Date
☑ Warmup Period
Warmup Start: Sun, Nov 27 2005 @ 12:00 AM
Sim. Begin: Sun, Nov 27 2005 @ 12:00 AM
Sim. End: Tue, Dec 27 2005 @ 12:00 AM

Clock Precision
0.001 ○ Second ○ Hour ● Minute ○ Day

Output Reporting
● Standard ○ Batch Mean ○ Periodic
Interval Length:
Number of Replications: 1

Disable
□ Animation □ Cost □ Array Export □ Time Series

At Start
□ Pause
□ Display Model Notes

General
☑ Adjust for Daylight Savings
□ Generate Animation Script
□ Common Random Numbers
□ Skip Resource DTs if Off-shift

General Options & Settings

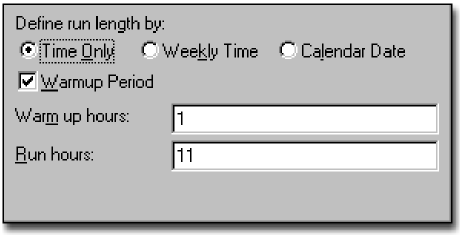
Output Path Contains the path of the output file (the name of the file is automatically created using the name of the model). ProModel records all statistical output in this file for your analysis.

If this field is left blank, the default path is used, which points to the Output directory in the ProModel directory.

When you change the default path, be sure the path you enter actually exists, or an error will occur at run time.

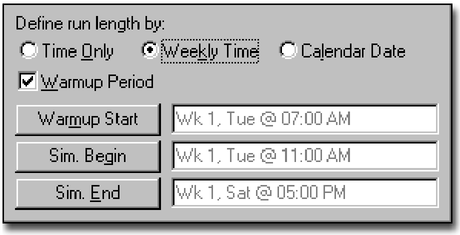
Define Run Length by ProModel allows you to define a run length for you model based on the duration you will test.

- **Time Only** The total run length in hours.



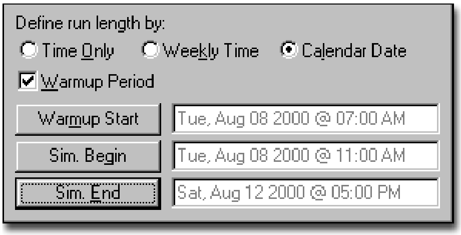
Define run length by:
● Time Only ○ Weekly Time ○ Calendar Date
☑ Warmup Period
Warm up hours: 1
Run hours: 11

- **Weekly Time** The total run length by week, day, and time.



Define run length by:
○ Time Only ● Weekly Time ○ Calendar Date
☑ Warmup Period
Warmup Start: Wk 1, Tue @ 07:00 AM
Sim. Begin: Wk 1, Tue @ 11:00 AM
Sim. End: Wk 1, Sat @ 05:00 PM

- **Calendar Date** The total run length by a specific calendar date and time.



Define run length by:

○ Time Only ○ Weekly Time ● Calendar Date

☑ Warmup Period

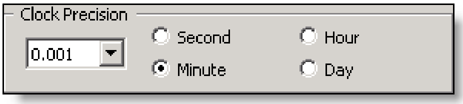
Warmup Start: Tue, Aug 08 2000 @ 07:00 AM

Sim. Begin: Tue, Aug 08 2000 @ 11:00 AM

Sim. End: Sat, Aug 12 2000 @ 05:00 PM

- **Warm-up Period** The amount of time to run the simulation before collecting statistics. Usually this is the amount of time it takes for the model to reach steady state. The warm-up period uses the same units as the run length.

Clock Precision Select the clock precision value from the drop down box. Then click the radio button for the clock units you want to display at run time.



Clock Precision

0.001

○ Second ○ Hour

● Minute ○ Day

Please note

The maximum run length depends on the clock precision and time unit selected. The following table shows the maximum run hours for each precision setting.

	CLOCK PRECISION			
Time Unit	**.01**	**.001**	**.0001**	**.00001**
Seconds (sec)	11,930 hrs	1,193 hrs	119 hrs	11 hrs
Minutes (min)	715,827 hrs	71,582 hrs	7,158 hrs	715 hrs
Hours (hr)	42,949,672 hrs	4, 294,967 hrs	429,496 hrs	42,949 hrs
Days (day)	1,030,792,128 hrs	103,079,208 hrs	10,307,904 hrs	1,030,776 hrs

The following table shows the default precision selections based on the time unit selected in the General Information dialog.

	DEFAULT CLOCK PRECISION SELECTIONS		
TIME UNIT	**Clock Unit**	**Clock Precision**	**Maximum Run**
Seconds (sec)	Seconds	.01	11,930 hrs
Minutes (min)	Minutes	.001	71,582 hrs
Hours (hr)	Hours	.001	4,294,967 hrs
Days (day)	Hours	.001	1,030,792,128 hrs

Disable: Animation Use this option to improve run-time speed by shutting down the animation. If you choose to disable animation, you cannot choose the Generate Animation Script option.

Disable Array Exports This option disables the exporting of data to external arrays. This is helpful for skipping the array exporting at the end of simulation, which may save time if the data being exported is large.

Disable Cost When you select this option, ProModel disables all cost information collection for in the model.

Disable: Time Series Use this option to improve run-time speed by foregoing time series statistics. This also saves disk space used to store the collected statistics.

At Start: Pause To pause the simulation at the start of the run, use this option. The simulation will pause until you select Resume from the Simulation menu in the Simulation module.

At Start: Display Notes Use this option to display the notes from the General Information dialog at the beginning of the simulation.

General: Adjust for Daylight Saving This option is only available when you have defined Run Length by Calendar Date. When checked, the simulation clock will account for the hour shift of daylight saving time in April and October.

General: Generate Animation Script When checked, the simulation will generate a 3D animation script for use with ProModel Corporation's 3D Animator application. The animation script records the graphical events of the simulation for later 3D rendering in 3D Animator.

General: Common Random Numbers This feature is a variance reduction technique used primarily when running multiple scenarios, each with multiple replications. It is intended to help reduce the number of replications required to determine the statistical significance of differences between scenario results. When enabled, common random numbers will ensure that the sequence of starting seed values for each stream in a set of replications for one scenario is identical to the starting seed values for corresponding replications in every other scenario.

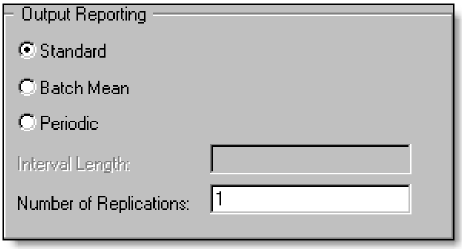
For example, if you have this option disabled, the match between starting seed values for each stream in corresponding replications for each scenario will not be guaranteed. The net effect of this is that you may have to run more replications to get the same confidence interval that you are able to obtain by running with common random numbers.

When common random numbers are enabled, you can theoretically establish the statistical significance of differences in output results between multiple scenarios with fewer replications than by not enabling common random numbers. Specifically, it reduces the amount of variation in differences between outputs of corresponding replications by ensuring that these differences are due to true differences in performance and not to differences due to randomness of the streams.

For more information on random number streams, see "Using Random Number Streams" on page 266.

General: Skip Resource DTs if Off-shift When checked, the simulation will ingnore resource downtimes if the resource is off-shift.

Output Reporting Options



Output Reporting

● Standard

○ Batch Mean

○ Periodic

Interval Length:

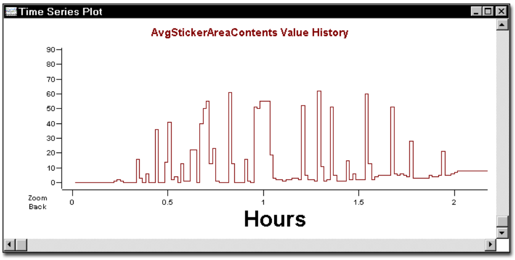
Number of Replications: 1

Standard When you select Standard output reporting, ProModel collects output statistics for one or more replications. No interval length can be specified when using this option. From the output program, statistics can be viewed for each replication; although, by default they are displayed as an average over all the replications.

Batch Mean The method of batch means, or interval batching, is a way to collect independent samples when simulating steady-state systems as an alternative to running multiple replications. The advantage over running multiple replications is that the warm-up period runs only once. When you select Batch Mean output reporting, the output statistics are collected for each time interval indicated in the Interval Length field. The number of intervals is determined by dividing the run length by the interval length. The interval length may be an expression but will only be evaluated once at model translation, so it is always a fixed interval. The Number of Replications edit field is not used when this option is selected since it replaces the need for running multiple replications.

Example

The Content History graph below shows that the contents of Lathe1 varied throughout the simulation. The interval length was set to 0.5 hours and the simulation ran a little over 3 intervals (1 1/2 hours). The average or mean of the intervals were 2.7, 3.09, and 3.15, so the average of the interval averages is the batch mean or 2.98.



Periodic Useful primarily in terminating or non-steady state simulations where you are interested in the system behavior during different periods (e.g., peak or lull periods) of activity.

When you select Periodic output reporting, the output statistics are collected by period where the length of a period is defined in the interval length field. The interval length may be an expression but will only be evaluated once at model translation, so it is always a fixed interval. To define unequal intervals, see “Customized Reporting” on page 352.

You may gather statistics for a periodic report over multiple replications. From the output program, you can view each replication averaged over the periods, each period averaged over the replications, or the pooled average (the average for each period averaged over all of the replications).

Example

The content history table below shows how the contents of Lathe1 varied throughout the simulation and from one replication to the other. The interval length was set to 0.5 hours and the simulation ran a little over 3 periods (11/2 hours) with two replications. The results are shown below.

Periodic Output	**Period 1**	**Period 2**	**Period 3**
Replication 1 Avg.	2.70	3.09	3.15
Replication 2 Avg.	2.20	3.13	3.01
Pooled Average	2.45	3.11	3.08

Interval Length Enter the interval length as the number of time units for each interval or period. The interval length may be an expression but will only be evaluated once at model translation, so it is always a fixed interval. To define unequal intervals, see “Customized Reporting” on page 352. The time unit is defined in the General Information dialog. The interval length need only be specified when using Batch Mean or Periodic reporting.

Number of Replications Enter the number of replications you want the simulation to run in this field. Number of replications only needs to be specified when using Standard or Periodic reporting.

Running a Specific Replication

ProModel allows you to run a specific replication in order to produce time series graphs in the Output Program. To run a specific replication, enter the “@” symbol in the “Number of Replications” box followed by the replication number. For example, “@5” will run only the fifth replication.

Customized Reporting

For customized reporting, you may want to take advantage of the following statements: REPORT, RESET STATS, and WARMUP. See “Report” on page 542, “Reset Stats” on page 545, and “Warmup” on page 579 for syntax and examples.

REPORT A general statement called from any logic. When the statement is called, a full set of statistics is saved to be viewed as a snapshot report in the output processor. Optionally, the statement allows you to reset the statistics after the report is saved, giving you a batch or period for any time interval you define.

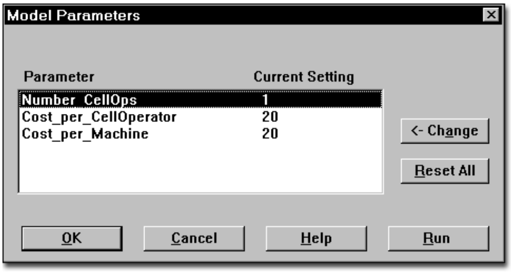
RESET STATS A general statement called from any logic and generally used in connection with the REPORT statement. When this statement is used, all statistics are set to zero. The output database is not erased (see WARMUP).

WARMUP A general statement called from any logic. When this statement is used, all statistics are set to zero and the output database is erased. Use this statement in conjunction with the WAIT UNTIL statement to wait until specific parameters in the system being modeled are at a steady state or other conditions are appropriate to declare the warm-up period over.

Model Parameters & Scenarios

Model Parameters

The Model Parameters dialog box allows you to modify the current settings for the Run-Time Interface (RTI) parameters defined in the macros module. This provides a convenient interface for making model changes without using the Build modules. To define and run multiple scenarios using RTI parameters, select the Scenarios option from the Simulation menu (see discussion on Scenarios later in this section). Model parameter settings are saved with the model for future use.



Parameter	Current Setting
Number CellOps	1
Cost_per_CellOperator	20
Cost_per_Machine	20

Parameter The name of the macro defined in the model as the RTI (Run-Time Interface) parameter. The parameter name does not need to be the same name as the macro.

Current Setting The current setting of the parameter.

Change Allows you to change the current setting of the parameter.

Reset All Resets all parameters to the default RTI setting defined in the Macros module.

Run Runs the model with the defined current model parameter settings.

How to define an RTI parameter:

1. Choose **More Elements** from the **Build** menu.
2. Choose **Macros...**
3. Type the macro name, choose the RTI button, and select Define.
4. Define the Parameter Name.
5. Enter the Prompt (optional).
6. Select the parameter type, Unrestricted Text or Numeric Range.
7. If defining a Numeric Range, enter the lower and upper boundary for the range.
8. Click **OK**.
9. Use the macro ID in the model (e.g., operation time, resource usage time, etc.).

Please note

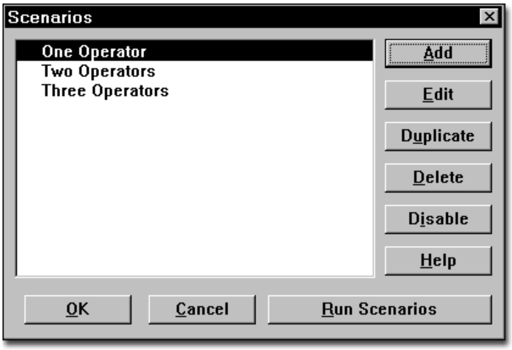
For more information on RTI, see "Run-Time Interface" on page 242.

Scenarios

ProModel gives you the option of defining several scenarios for a model using RTI parameters specified for the model. A scenario is a set of run-time parameters with settings defined by the user. Using scenarios allows you to alter various model parameters to run a series of "what-if" scenarios without changing the model directly.

Scenarios can also be helpful for allowing other users of you model, who may not have experience using model logic, to make changes to the model through RTI parameters.

Scenarios are saved with the model for future use.



Add Opens the Scenario Parameters dialog box to add a scenario.

Edit Opens the Scenario Parameters dialog box to edit an existing scenario.

Duplicate Duplicates the selected scenario and opens the Scenario Parameters dialog box, allowing you to give the newly created scenario a name and edit its data.

Delete Deletes the selected scenario.

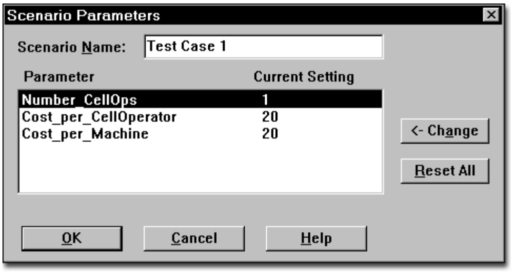
Disable/Enable Disables or enables the selected scenario.

Run Scenarios Runs the model with the defined scenarios. When running several scenarios, clicking on the Abort button during translation will terminate all scenarios instead of just the current scenario.

Scenario Parameters Dialog

The Scenario Parameters dialog box is displayed when choosing Add, Edit, or Duplicate from the

Scenarios dialog box. This allows you to control the specifics of a scenario.



Scenario Name: Test Case 1

Parameter	Current Setting
Number_CellOps	1
Cost_per_CellOperator	20
Cost_per_Machine	20

Scenario Name The name of the scenario. This name is chosen by the user and may contain any alphanumeric characters.

Parameter The name of the parameter as defined in the RTI (Run-Time Interface) definition dialog box invoked from the Macro editor.

Current Setting The current setting of the parameter.

Change Allows you to change the current setting of the parameter.

Reset All Resets all parameters to their default settings.

Please note

Only macros defined as RTI parameters will be displayed as parameters for the scenario. (See "Macros" on page 241.)

How to define a scenario:

1. Choose **Scenarios...** from the **Simulation** menu.

2. Choose the **Add** button.

3. Define the scenario name by typing text in the Scenario Name box.

4. Double click on the Parameter to bring up the parameter dialog box or select the Parameter and click the **Change** button.

5. Type the text in the box in the Parameter dialog box.

6. Select **OK** in the Parameter dialog box.

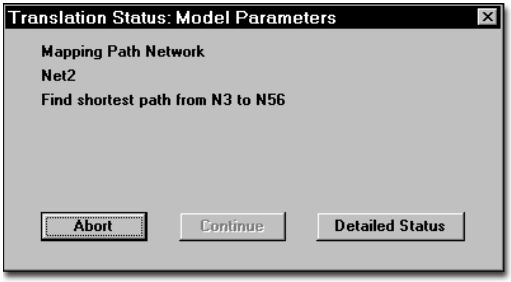
7. Repeat for every parameter desired.

8. Select **OK** in the Scenario Parameters dialog box.

Running the Simulation

Choosing Run begins the simulation of the current model. The model is automatically saved in a file called autosave.mod. Therefore, if the simulation is terminated abnormally, ProModel will ask you if you would like to load the latest autosave.mod file when you re-open ProModel. If you choose Save & Run, the model will be saved in the AUTOSAVE.MOD file as well as the <model name>.mod file.

Once you choose the Run option, a translation status window appears, showing which data is currently being translated. This gives you the option to abort the simulation run at any time. As shown below, the path networks are being mapped.



Mapping Path Network
Net2
Find shortest path from N3 to N56

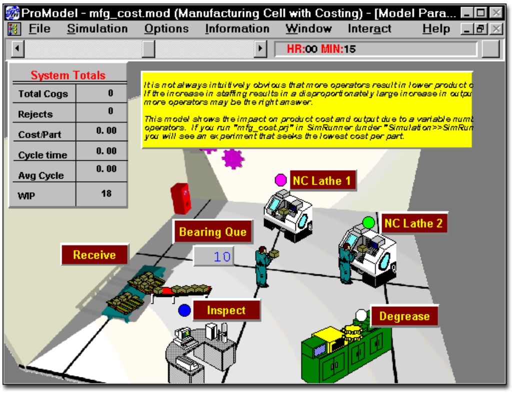
Abort Allows you to cancel model translation at any time during translation.

Continue Allows you to continue model translation when a warning message is displayed.

Detailed Status Allows you to view more specific information about the data in translation.

Run-Time Menus & Controls

Once a simulation begins, a new menu bar appears at the top of the screen with selections for controlling and interacting with the simulation. As shown below, these menu items appear above the animation speed control bar and simulation clock.



System Totals	
Total Cogs	0
Rejects	0
Cost/Part	0.00
Cycle time	0.00
Avg Cycle	0.00
WIP	18

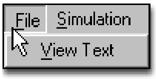
It is not always intuitively obvious that more operators result in lower product c
If the increase in staffing results in a disproportionately large increase in outpu
more operators may be the right answer.

This model shows the impact on product cost and output due to a variable numb
operators. If you run "mfg_cost.prj" in SimRunner (under "Simulation>>SimRun
you will see an experiment that seeks the lowest cost per part.

Each of the menu selections and the tools for controlling the animation are described in the remainder of this chapter.

Run-Time File Menu

The Run-Time File menu contains only one selection, View Text.



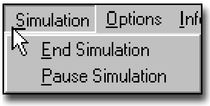
View Text Choose this option to bring up a window with a text listing of the current model. This feature is extremely useful for debugging and verifying models.

Please note

You may switch back and forth between a full size View Text window and the animation screen by *choosing the desired window from the Window menu option.*

Run-Time Simulation Menu

The Run-Time Simulation menu has two options: End Simulation and Pause/Resume Simulation.

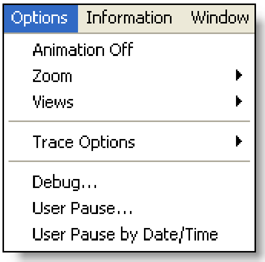


End Simulation Choose this option to end the simulation. You will then be prompted to collect statistics or return to the model editor without collecting statistics. If running multiple scenarios, End Simulation will terminate all scenarios.

Pause/Resume Simulation Choose Pause Simulation to pause the simulation for an indefinite amount of time. With the simulation paused, you may begin a trace, zoom in or out, set up the next pause, examine different locations in the model, or interact with the model in a number of other ways. Choose Resume Simulation when the simulation is paused to continue running the simulation.

Run-Time Options Menu

The run-time Options menu has several selections that allow you to interact with the simulation while the model is running. These options are described in the following pages.



Animation Off/On Turns the animation on or off. Off greatly speeds up the simulation. Animation speed may also be set with the ANIMATE statement. (See “Animate” on page 442 for information).

Zoom Zooms in or out on the animation.

Views Allows you to select a view, which you have defined while building the model. This option is only available if you have previously defined a view.

Trace Options Lists events as they happen during a simulation. This listing may be Step by Step, Continuous, or Filtered.

Debug Brings up the Debugger Options Dialog box for debugging the model.

User Pause Allows the user to enter a simulation clock time for the simulation to pause.

User Pause by Date/Time Allows the user to enter a calendar date and simulation clock time for the simulation to pause. This option is only available if you chose "Calendar Date" as the Run Length in the Simulation Options dialog.

Debug Option

Debugging ProModel Logic

The Debugger is a convenient and efficient way to test or follow the processing of any logic defined in your model. The debugger is used to step through logic one statement at a time and examine variables and attributes while a model is running.

Before discussing the details of the Debug option, it is important to understand the following terms:

Statement A statement causes ProModel to take some action or perform some operation. This includes statements such as GET, JOIN, and SPLIT AS. (See "Statements and Functions" on page 439 for more information).

Logic Logic refers to the complete set of statements defined for a particular process record, downtime event, initialization logic, or termination logic for a simulation.

Thread A thread is a specific execution of any logic. A thread is initiated whenever a logic needs to be executed. This can be an entity running through an operation logic, the initialization logic, a resource running a node logic, a downtime logic, or any other logic. Note that the same logic may be running in several threads at the same time. For example, three entities of the same type being processed simultaneously at the same multi-capacity location would constitute three threads.

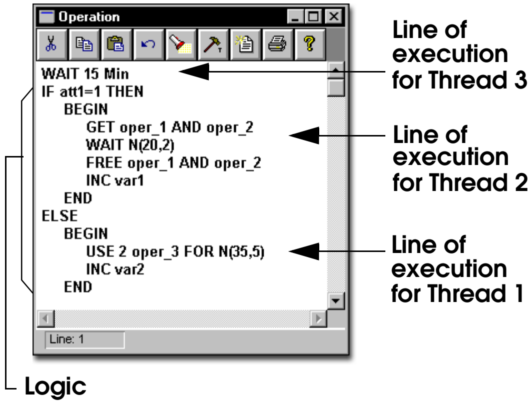
A thread or logic execution can be suspended by any statement, causing simulation time to pass (e.g., GET Res1, WAIT 5, etc.). After such a statement completes its task, the thread is resumed. During the time a thread is suspended, other threads may be initiated, suspended, resumed, or completed. This is called thread switching.

Please note

Even though several threads can execute the same logic at the same time in the simulation, the simulation processor can only process them one at a time. So there is really only one current thread while all other threads are suspended (either scheduled for some future simulation time, or waiting to be executed after the current thread at the same simulation instant).

Example

To better explain the above concepts, consider the following operation logic for a multi-capacity location.


Operation
Line of execution for Thread 3
Line of execution for Thread 2
Line of execution for Thread 1
Line: 1
Logic

```
WAIT 15 Min
IF att1=1 THEN
   BEGIN
      GET oper_1 AND oper_2
      WAIT N(20,2)
      FREE oper_1 AND oper_2
      INC var1
   END
ELSE
   BEGIN
      USE 2 oper_3 FOR N(35,5)
      INC var2
   END
```

The logic includes all statements shown on previous page. Let's assume that there are three different entities currently executing this operation logic. Each executing entity constitutes a thread. A possible scenario for this case is the following: Thread 1 is an entity using two units of resource Oper_3 for N(35,5) minutes. Thread 2 is an entity waiting for resource Oper_1 to become available. The last thread is a different entity which has completed 10 minutes of the 15 minute wait executed at the beginning of the logic. Note that two entities arriving at a multi-capacity location at nearly the same time could both execute the same WAIT or USE time in the logic, only in different threads.

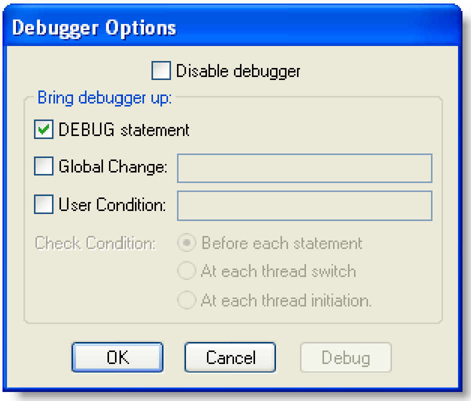
In general, for a logic block containing statements that pass simulation time, any number of threads can wait for the required simulation time to elapse, corresponding to each time elapsing statement. These threads are scheduled to resume at some future simulation time.

There can also be many threads (any number of threads corresponding to each of the time elapsing statements within the logic) which have completed their waiting time, but await their turn to continue execution. These threads are on hold because the simulation engine is busy with another thread scheduled for the same simulation time. There is only one thread executed at any real time instant by the simulation engine.

The debugger window will display a unique identification number for the current, active thread. This thread ID number will help you differentiate between different instances of the same logic block while you are debugging your models.

Debugger Options Dialog Box

The Debugger Options dialog box allows the user to specify when to display the Debugger dialog box during the simulation run.



Debugger Options

□ Disable debugger

Bring debugger up:

☑ DEBUG statement

□ Global Change:

□ User Condition:

Check Condition: ● Before each statement
○ At each thread switch
○ At each thread initiation.

Disable debugger Disables the debugger completely. By default the debugger is enabled. Running the model with the debugger disabled increases the run speed. When running multiple replications or scenarios, or when the animation is disabled, the debugger will automatically be disabled.

DEBUG statement Displays the Debugger dialog box every time a DEBUG statement is encountered in an enabled process while running the simulation. See “Debug” on page 465 for more information.

Global Change Displays the Debugger dialog box every time a global change occurs to a specified variable or array. The Debugger dialog box shows the original and new value of the element. Only one global name can be specified in this box.

User Condition Displays the Debugger dialog box when a defined user condition written as a Boolean expression becomes true, for example, when Var1=5. Only one expression can be specified in this box, although several conditions can be tested by using the OR operator. See "Boolean Expressions" on page 410 for more information.

Check Condition Allows the user to define how often to check the user condition. The options include:

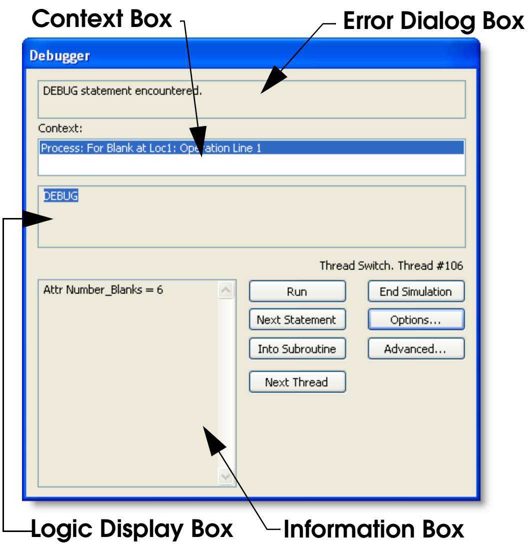
- **Before each statement** The condition, such as Var1=1, will be checked before each statement is executed. This option is the most precise way to tell exactly when the user condition becomes true, but it slows down the simulation the most.
- **At each thread switch** The condition, such as Var3>17, will be checked only if a statement from a different thread follows the current statement being executed.
- **At each thread initiation** The condition, such as Att1=5, will be checked only if the next statement to be executed is the first statement in a thread (the first statement in a thread is also the first statement of a logic).

Debug button Pressing the Debug button displays the Debugger dialog box before the next statement executes.

OK Closes the Debugger Options dialog box and continues to run the simulation model.

Debugger Dialog Box

The Debugger can be used in two modes: Basic and Advanced. The Basic Debugger appears initially with the option of using the Advanced Debugger. The Basic Debugger dialog box is shown below:



DEBUG statement encountered.

Context:

Process: For Blank at Loc1: Operation Line 1

DEBUG

Thread Switch. Thread #106

Attr Number_Blanks = 6

Error Display Box Displays the error message or reason why the Debugger dialog box is displayed, such as the User-Condition becoming true.

Logic Display Box Displays the statements of the current logic being executed.

Context Box Displays the module, operation, and line number (in which the debugger stopped) in the Information box.

Information Box Displays local variables and entity attributes with non-zero values in the Information box.

End Simulation Choose this option to terminate the simulation. This will prompt you about collecting statistics.

Run Continues to run the simulation, but still checks the debugger options selected in the Debugger Options dialog box.

Next Statement Jumps to the next statement in the current thread. If the last statement executed suspends the thread (e.g., the entity is waiting to capture a resource), another thread meeting the debugger conditions may be displayed as the next statement.

Next Thread Brings up the Debugger at the next initiated or resumed thread.

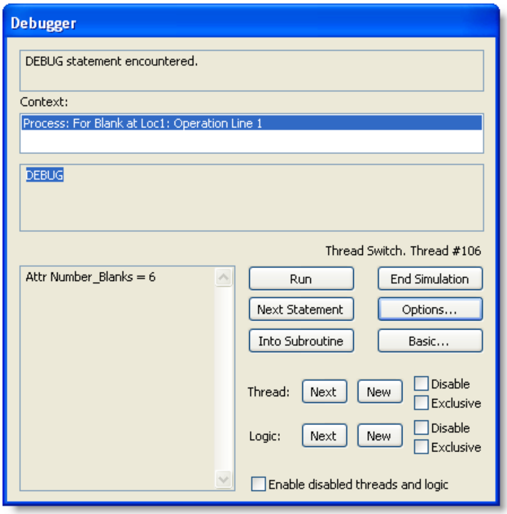
Into Subroutine Steps to the first statement in the next subroutine executed by this thread. Again, if the last statement executed suspends the thread, another thread meeting debugger conditions may be displayed first. If no subroutine is found in the current thread, a message is displayed in the Error Display box.

Options Brings up the Debugger Options dialog box. You may also bring up this dialog box from the Simulation menu.

Advanced Changes the Debugger to Advanced mode, provides additional options discussed next.

Advanced Debugger Dialog Box

The Advanced Debugger contains all options in the Basic Debugger plus a few advanced features.



Debugger

DEBUG statement encountered.

Context:

Process: For Blank at Loc1: Operation Line 1

DEBUG

Thread Switch. Thread #106

Attr Number_Blanks = 6

Next (Thread) Jumps to the next initiated or resumed thread. This button has the same functionality as the Next Thread button in the Basic debugger.

New (Thread) Jumps to the next initiated thread.

Disable (Thread) Temporarily disables the debugger for the current thread (see also enable).

Exclusive (Thread) The debugger displays the statements executed within the current thread only. When the thread terminates, the exclusive setting is removed.

Next (Logic) Jumps to the next initiated or resumed thread that is not executing the same logic as the current thread.

New (Logic) Jumps over any resumed threads to the next initiated thread not executing the same logic as the current thread.

Disable (Logic) Temporarily disables the debugger for all threads executing the current logic (see also enable).

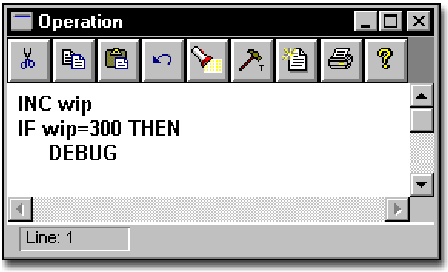
Exclusive (Logic) The debugger displays only the statements executed in any thread executing the current logic.

Enable disabled threads and logics Enables the threads and logics which were disabled previously.

Debugger Options Examples

Debug Statement Example

A simulation model demonstrates a proposed flexible manufacturing system which produces castings. A variable, WIP, is used to track the work in process for the system. Suppose we want to display the Debugger when the variable, WIP, reaches a value of 300. We could place an IF...THEN statement including DEBUG after the statement incrementing the variable, WIP, as shown below:



```
INC wip
IF wip=300 THEN
   DEBUG
```

By checking the box next to the DEBUG statement in the Debugger Options dialog, the Debugger is displayed when the variable, WIP, reaches 300.

Global Change Example

Suppose we want to know when the variable, COUNT, is incremented. We would check the box to the left of Global Change and type COUNT in the Global Change field. This will display the Debugger dialog box each time COUNT changes. It will also display the previous and changed value of COUNT.

User Condition Example

Suppose we want to follow a casting through the entire system (i.e., from when an entity enters the system to when it exits). We would set an attribute equal to a unique number in the arrival logic for a single casting (Att1=3). In other words, only one casting in the system should have Att1=3. We would then check the User Condition box and enter the condition as "Att1=3." We would then select At Each Thread Initiation as the Check Condition. The debugger is displayed each time a particular customer initiates a new thread. For example, the debugger may display the following information:

Debugger syntax

```
Casting @ loc1: GET Oper_1
    WAIT N(5,3)
    INC Var1
Casting @ loc2: JOIN 1 Fixture
...
...
...
```

Trace Options

A trace is a list of events occurring over the course of a simulation. For example, a trace statement might state "EntA arrives at Loc1, Downtime for Res1 begins." A trace listing also displays assignments, such as variable and array element assignments. A trace listing of a simulation run may be viewed in several ways through the trace options provided.

Trace Mode

A trace listing is generated in one of three modes, Step, Continuous, or Filtered.

Trace Off

Select this option to discontinue a current trace.

Trace Step

Select this option to step through the trace listing one event at a time. Each time you click the left mouse button, the trace will advance one event. Clicking and holding the right mouse button while in this mode generates a continuous trace.

Trace Continuous

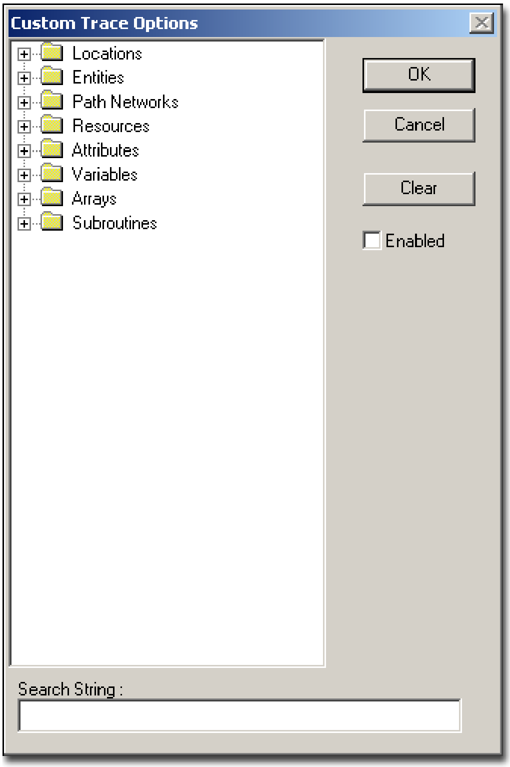
Select this option to write the trace continuously to the output device selected from the Trace Output submenu. This is useful when you do not know exactly where to begin or end the trace. Clicking and holding the right mouse button stops the trace until you release the button.

Filtered Trace...

When you view the Trace during simulation, you see a list of all the events that occur as your model is running.

Although it may be helpful to view every event in the simulation, there are many times when you will want to focus on just the events that affect specific elements (locations, resources, variables, etc.) of your model.

The Trace feature contains an optional, custom filter, which allows you to pick the elements of your model, or custom text string, that you wish to view in the Trace window.



From the Custom Trace Options dialog, you may choose which elements of your model you want displayed in the Trace window.

The Custom Filter works by showing the specified elements in the trace according to the following:

- Specified elements found in the main heading of the Trace will be displayed along with the entire contents of the sub-headings.
- Specified elements found in a sub-heading will only display the main heading and the sub-heading with the specified element. All other sub-headings of the main heading, which do not contain the specified elements, will not be displayed.

The Enable button in the Custom Trace Options dialog must be checked for the Trace Filter to take effect.

Output to File

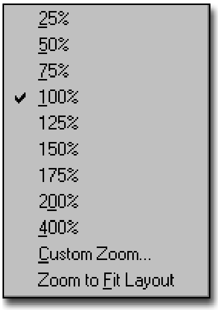
Select this option to send the trace listing to a text file. Trace statements are automatically written to a <model name>.TRC file.

Animation Options

In addition to the debug and trace options, animation options allow you to control the animation screen.

Animation Off Choose this option to temporarily suspend the animation. To resume the animation, select this option again. (The selection automatically changes to “Animation On.”) Note that running with animation off greatly increases the run speed, especially for models with a large amount of graphic detail. To increase the run speed to an even faster rate, check the Disable Animation in the Simulation Options before running the model (see “Animation Options” on page 363).

Zoom Select this option to Zoom in or out on the animation. When this option is selected, you may choose a preset zoom level, enter your own zoom level, or choose Zoom Full to fit the entire animation on one screen. The Zoom function zooms to the center of the screen. If the zoom factor causes the model layout to appear outside the layout window, the zoom function will automatically pan to show at least part of the layout.



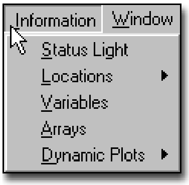
To zoom in a specific area of the simulation layout, press and hold the CTRL key, then click and drag the mouse to create a rectangle around the are in which you would like to zoom. Release the mouse, then the CTRL key. The layout will zoom in on the selected area.

Views Click on this menu item to display the available views. Select the desired view.

User Pause Choose this option to enter a time for the simulation to pause. The proper format for specifying a user pause is hh:mm:xx where hh represents hours, mm represents minutes, and xx represents hundredths of a minute.

Run-Time Information Menu

The Run-Time Information menu allows you to see the status of locations in two different ways. In addition, you may view the current state of all variables and array elements. Each of these options is defined in the following pages.



Status Light Select this option to bring up the Status Light Legend.

Locations Select this option and choose a location to view an information box with real time information about the location. Information for all locations may also be displayed.

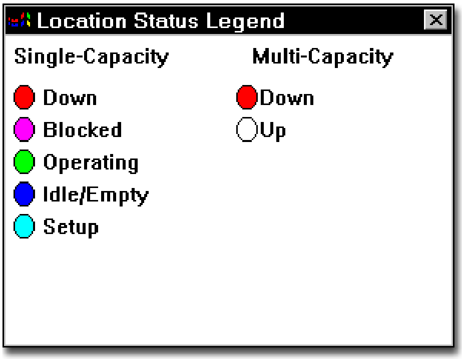
Variables Select this option to show the current state of all real and integer global variables.

Arrays Select this option to show the current value of all cells for arrays of up to three dimensions.

Dynamic Plots Allows you to graphically monitor the performance of model elements during run time and store statistical data in an Excel® spreadsheet.

Location Status Legend

The Location Status Legend shows the different colors of a location status light and the meaning of each color. Single capacity locations may be in any of several states, while multi-capacity locations appear only as up or down. This window may remain open during the simulation.



Location Status Legend

Single-Capacity	Multi-Capacity
🔴 Down	🔴 Down
🟣 Blocked	○ Up
🟢 Operating	
🔵 Idle/Empty	
● Setup	

Dynamic Plots

Dynamic plots allow you to select certain metrics for various model elements and observe value changes for those metrics dynamically as the model runs. Configurations of one or more plot windows can be saved and later retrieved to quickly view a customized set of graphs.

Features include:

- Up to six elements many be graphed on the same chart at the same time.
- More than one chart may be active at a time.
- Charts can be resized.
- Plots display a meaningful scale for both axes.
- Improved customization for graph appearance
- Chart settings may be saved to use over from one simulation run to the next.
- Dynamic plots work with multiple replications and scenarios.

Basic Operation

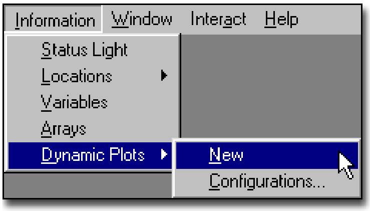
Please note

The Dynamic Plot dialog is only accessible during simulation run-time. If you will be creating saved chart views, you may wish to pause the simulation, define Chart views, then resume the simulation.

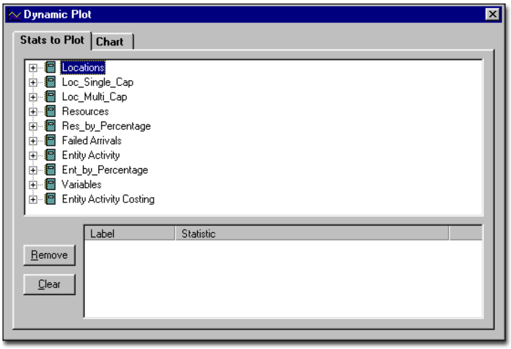
Starting Dynamic Plots

Dynamic plots are set up using the **Information | Dynamic Plots** menu during simulation. Under

Dynamic Plots you will find two sub-items, **New** and **Configurations…**



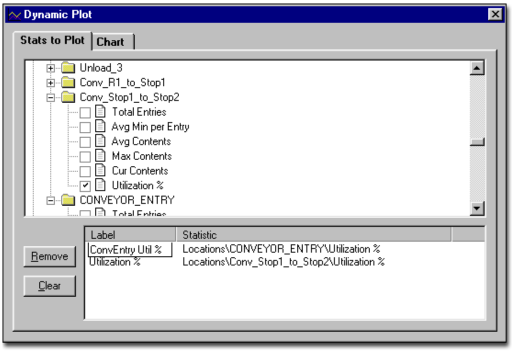
New opens a window similar to the one below.



Setting up a Plot

Tree View

The tree view represents a hierarchical list of all plottable items in the model. Expanding and collapsing of tree items works in a manner consistent with standard Windows tree controls, including, but not necessarily limited to, double-clicking on a parent item, clicking on plus or minus graphic next to a parent item, and using the plus and minus keys on the number pad of most keyboards. When a subtree is fully expanded, individual plottable items are shown, as below:



Label	Statistic
ConvEntry Util %	Locations\CONVEYOR_ENTRY\Utilization %
Utilization %	Locations\Conv_Stop1_to_Stop2\Utilization %

Each plottable item displays a checkbox to its left. When checked, the item is added to the list of currently plotted items below the tree view and the chart view begins to plot the value of the item. When unchecked, the item is removed from the list of currently plotted items, and the item is removed from the chart view. Items can also be added or removed from the chart view by double-clicking the item. If the item was not checked, this will check the box next to the item, add it to the current item list and the chart view, and switch immediately to the chart display. If the item was checked, this will uncheck the box next to the item, remove it from the current item list and the chart view, and switch immediately to the chart display.

The maximum number of statistics that can be plotted on a single chart is six. When six items are being charted and you attempt to add an additional item by either clicking the box next to the item or by double-clicking the item, nothing will happen and no changes to the chart will take place.

Statistic List

The item list below the tree view contains the items currently being graphed, arranged into two columns: a label and a statistic name. The label initially consists of only the text directly right of the item icon in the tree view, while the statistic name is composed of the names of the items ancestor nodes and the name of the item node itself. For example, for the "Avg Min per Entry" item under the Input location, the label would initially be set to "Avg Min per Entry" and the statistic name would be "Locations\Input\Avg Min per Entry."

Please note

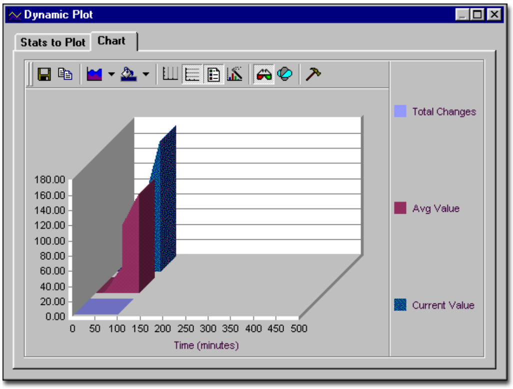
When you click on the Label portion of the item in the list to select it and then click again, the label becomes editable, and you can type over or edit the existing text. This text becomes the new label for the item, and will now be displayed in the legend in the chart view.

The **Remove** button removes the selected item, if any, from the list, unchecks it in the tree view, and removes it from the chart view. Clicking the **Clear** button does this for every item in the statistic list.

Clicking the dividing lines between headings and dragging left or right resizes the columns in the statistic list.

Chart View

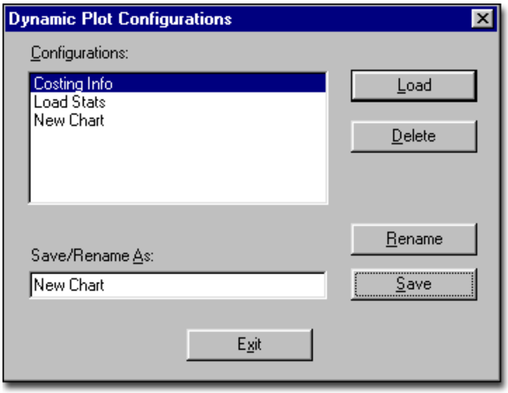
The chart view is where graphical representations of changing model elements are displayed. It consists of a chart area and, optionally, any combination of the following: toolbar, legend, x- and y-axis scales, x- and y-axis labels, vertical and horizontal gridlines, and a chart title. An example of a chart view is shown below.



The x-axis indicates time elapsed from the start of the simulation, measured in clock precision units specified in the Simulation Options dialog. The y-axis measures values of the items being plotted. Both axes are re-scaled when necessary so that all data points fit within the chart area. When in the chart view, the maximize button in the title bar expands the chart itself to completely fill the area within the dialog, removing the “Stats to Plot” and “Chart” tabs and extra spacing. This may be helpful when you want to size the chart window as small as possible and still view the chart update. Clicking the minimize button will restore the chart to its normal state. The items being plotted are color-coded and the key is on the right-hand side of the window. Many of the chart options such as title and colors are customizable; simply click the right mouse button and select **Properties…** from the context sensitive menu options.

Dynamic Plot Configurations

Information | Dynamic Plots | Configurations... is disabled if no saved configurations exist for the model and no dynamic plot windows are open. If enabled, selecting it opens a dialog box like the one below. The dialog box consists of a list box containing the names of any saved dynamic plot configurations for the model, an edit field, and Load, Rename, Delete, Save, and Exit button.



If you select a configuration from the list and click **Load**, the indicated saved configuration of chart(s) will be loaded, with all of the previous settings as far as selected data items, chart style, colors, fonts, visible elements, screen position and size, etc. If any of the previously selected data items no longer exist in the model, they will simply not be displayed. For example, if a statistic had been selected for a location that was later deleted from the model, it will no longer be selected.

Double-clicking on a configuration in the list has the same effect as selecting it and clicking the **Load** button.

The **Delete** button removes the selected configuration from the list and deletes the associated data from the model.

The **Rename** button changes the name of the currently selected configuration to the text in the **Save/Rename As** edit field.

Clicking the **Save** button saves all settings of any open dynamic plot windows, including selected data items, chart style, colors, fonts, visible elements, screen position and size to a configuration with the name specified in the edit field. If the name matches one of the configurations in the configuration list, this data will replace the previous data for that name.

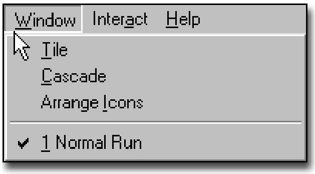
The **Save** button does not actually save data into the model file. Rather, it creates or modifies a set of configuration data associated with a particular configuration name in memory and tags the model as being modified. If you exit the program, load a model, or execute the New command from the File menu you will be prompted to save the model. Using the **Delete** or **Rename** buttons in the configuration dialog also tags the model as modified.

Advanced Operation

To enhance your simulations and presentations you can turn your dynamic plots on and off using the **DynPlot “”** statement in your model logic. Predefine the statistics to be graphed and a chart name, then open your plot through subroutines or processing logic.

Run-Time Window Menu

The Run-Time Window menu allows you to rearrange windows and icons and select the active window. These functions are standard to all Windows applications.



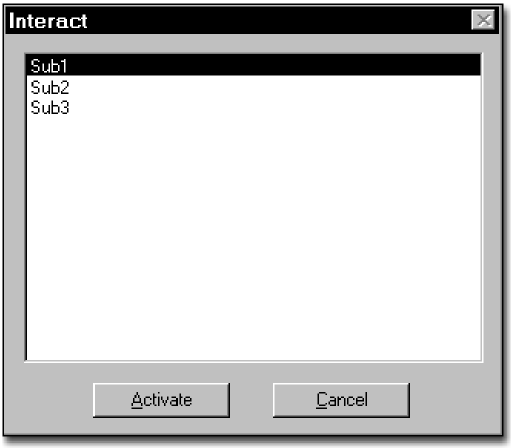
Tile Causes all open windows to fit in the available screen space. Windows hidden behind other windows become visible.

Cascade Causes all open windows to overlap such that the title bar of each window is visible.

Arrange Icons Causes all icons representing iconized applications or windows to be arranged neatly along the bottom of the screen.

Run-Time Interact Menu

The Run-Time Interact menu displays the Interact dialog box. It allows you to execute interactive subroutines during run-time. Interactive subroutines are defined in the Subroutines edit table (see “Subroutine Editor” on page 246).

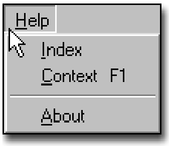


How to execute an interactive subroutine during run-time:

1. Select **Interact** from the run-time menu.
2. Select the identifier for the subroutine from the list box in the Interact dialog box.
3. Choose the **Activate** button.

Run-Time Help Menu

The Run-Time Help menu contains selections for accessing the ProModel on-line help system. It operates the same here as in the model editing functions.



Index... Choose this option to bring up the Main Help Index.

Context Choose this option to go directly to the help screen that corresponds to the active window. If no context sensitive help exists for the active window, the Main Help Index will appear.

About... Choose the option to display a message containing software version information.

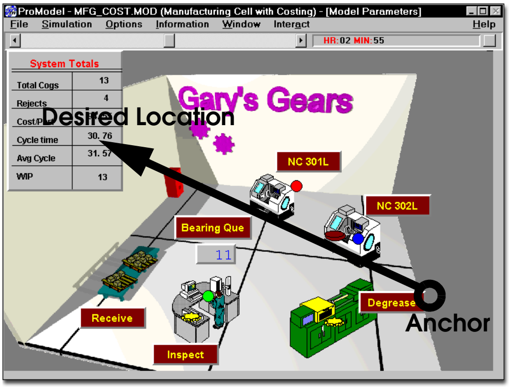
Run-Time Controls

In addition to the animation options discussed in the previous section, you may pan the animation screen in any direction, control the speed of the simulation, and change the format of the simulation clock display. These and other procedures are discussed in this section.

How to pan through the animation:

1. Press and hold the left mouse button anywhere on the animation screen. This point then becomes the "anchor point" by which the entire animation is moved.

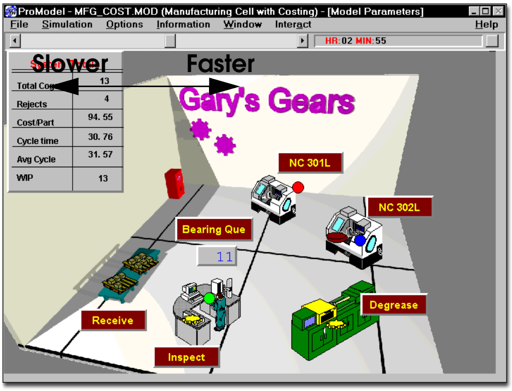
2. Drag and release the mouse where you desire the anchor point to be located. The following diagram illustrates this procedure.


ProModel - MFG_COST.MOD (Manufacturing Cell with Costing) - [Model Parameters]
File Simulation Options Information Window Interact Help
HR:02 MIN:55
Gary's Gears
Desired Location
NC 301L
NC 302L
Bearing Que
11
Receive
Inspect
Degrease
Anchor

System Totals	
Total Cogs	13
Rejects	4
Cost/Part	94. 55
Cycle time	30. 76
Avg Cycle	31. 57
WIP	13

How to control the simulation speed:

• Move the animation speed control bar to the left to decrease the simulation speed or to the right to increase the simulation speed.


ProModel - MFG_COST.MOD (Manufacturing Cell with Costing) - [Model Parameters]
File Simulation Options Information Window Interact Help
HR:02 MIN:55
Slower
Faster
Gary's Gears
NC 301L
NC 302L
Bearing Que
11
Receive
Inspect
Degrease

Total Cogs	13
Rejects	4
Cost/Part	94. 55
Cycle time	30. 76
Avg Cycle	31. 57
WIP	13

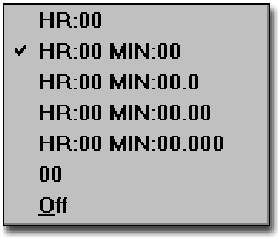
Please note

The speed of the animation may be altered by model logic with the use of the ANIMATE statement (See "Animate" on page 442).

How to change the format of the simulation clock display:

1. Click on the simulation clock button.

2. Select a format for the clock display.

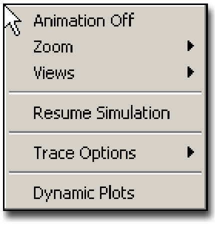


How to identify any location on the layout during the simulation:

- Click on a location while holding the CTRL key. (An identifier box appears and allows you to bring up the location information window for that location.)

Run-Time Right-Click Menu

Right clicking in the simulation window will bring up a right-click menu, which gives you easy access to several animation controls.



- **Animation Off** Turns off the animation, which makes the simulation run faster.
- **Zooms** Allows you to choose a level of zoom.
- **Views** Allows you to choose a view that you have previously defined. See "Views" on page 84 for information on defining views.
- **Pause/Resume Simulation** Allows you to toggle the simulation's pause on and off.
- **Trace Options** Lists events as they happen during a simulation. This listing may be Step by Step, Continuous, or Filtered.
- **Dynamic Plots** Opens the Dynamic Plots window.

Please note

Right clicking to bring up the right-click menu will pause the simulation as long as the right-click menu is open.

SimRunner

SimRunner takes your existing ProModel simulation models, evaluates them *for you*, then performs tests to find better ways to achieve the results you desire. Typically, most people use simulation tools to predict and improve a system's performance by modeling the actual location (e.g., a plant floor, a bank lobby, or emergency room) or abstract process (i.e., a logical process). Through testing various "what-if" scenarios, SimRunner can help you determine the most ideal way to conduct operations—we call this *optimization*.

When you conduct an analysis using SimRunner, you build and run *projects*. With each project, SimRunner runs sophisticated optimization algorithms on your model to help you optimize multiple factors simultaneously. For each project, you will need to provide SimRunner with a model to analyze or optimize, identify which input factors to change, and define how to measure system performance using an objective function. SimRunner can conduct two types of tests: Pre-Analysis (Statistical Advantage) and Simulation Optimization.

SimRunner Benefits

Using SimRunner will help you find *accurate* solutions for your modeling needs. No longer must you sit and experiment with what you think might work, only to find that your solution actually *interferes* with productivity in other parts of the model. SimRunner will help you locate true solutions by monitoring how changes affect each part of the model. In other words, SimRunner will *not* let you improve one area of your model at the expense of another—the results you get are beneficial to the *entire* model.

Starting a New Project

The following describes how to start and prepare a project for analysis. Remember that you must create and validate your model *prior* to analyzing or optimizing it in SimRunner.

How to select a model:

1. Click the **New** button on the button bar or select **New Project** from the **File** menu.

2. From the Open dialog, select the model file you wish to use.

3. Click **OK**.

4. Once you select a model, the model's name will appear in the Project Information dialog.

How to define the input factors:

1. Click on the **Inputs** button.

2. To select input variables, click next to each variable in the **Selected** column. SimRunner will mark the selected items.

Please note

SimRunner will optimize only single value RTI macros.

How to define the objective function (output variables):

1. Click the **Outputs** button.

2. In the Objective Function Setup dialog, select the category of the item(s) you wish to minimize or maximize.

3. Highlight an item in the list.

4. Select **Maximize**, **Minimize**, or **Custom** to select the type of optimization you wish to use for the variable.

5. Enter the **Weighting**. This number represents the importance of Maximizing or Minimizing the item. (The higher the number, the higher the importance.)

6. After you set the values for the item, click the right arrow to place the item in the selected list. To remove an item from the selected list, select the item and click the left arrow. (Alternatively, you may double click on an item to add it to or remove it from the list of selected variables.)

Stage one: Pre-Analysis

With the model built, the input factors selected, and an objective function defined, you are ready to conduct a Pre-Analysis. Also known as Statistical Advantage, the Pre-Analysis runs several tests to identify the initial bias (warm-up period), determine appropriate run-length to reach steady-state, find the number of replications necessary to ensure that each event occurs at least once, and locate the model averages.

Stage two: Simulation Optimization

Simulation Optimization is a multi-variable optimization that tries different combinations of input factors to arrive at the combination that provides the best objective function (output) value. Depending on the number of selected input factors and the complexity of the solution space, this process can take a long or short time. Optimizations with many factors and complex solutions take longer to run.

How to run an optimization:

1. Click on the **Optimize!** button or select **Start Optimization** from the **Project** menu.
2. Click **Play** on the dashboard to begin.

Chapter 10: Reports and Graphs

Once your model has been built, and the simulation run, you are ready to begin making important decisions about your real-world process based on your simulation's data. To help you make the most of the data collected during simulation, ProModel comes with a powerful, easy-to-use Output Viewer 3DR. This Output Viewer 3DR allows you to view your date numerically, in spreadsheet format, or graphically in a variety of charts.

Output Viewer 3DR

Output Viewer 3DR organizes and displays the data gathered during your model's simulation.

You can open Output Viewer 3DR from ProModel or from the Window's Start menu in the ProModel program group.

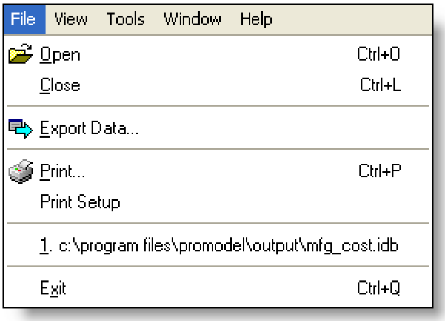
Output Viewer 3DR's Menu bar and Toolbar give you the controls necessary to create the reports and charts needed to help interpret your data.

Menu Bar

The Menu bar contains all of the tools necessary to view your simulation's data. The Menu bar is located just beneath the Output Viewer 3DR caption bar, and allows access to the following menus:

File Menu

The File menu allows you to open and close data files, export data, and print data.



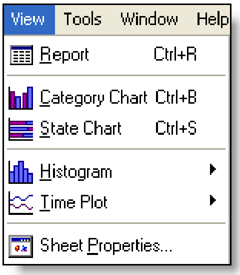
Output Viewer 3DR opens files with the .idb extension; however, .rdb files, which are generated by ProModel or Process Simulator can be converted to .idb files when they are opened using Output Viewer 3DR.

Data can be exported to a .csv (comma-delimited) file using the Export Data option. This allows you to later import the data into other data viewing programs, such as Excel

Data sets can also be printed using the print options in this menu.

View Menu

The View menu lets you specify how you want the data displayed.



Reports and Charts

The first five selections in the View menu display the output data in different ways:

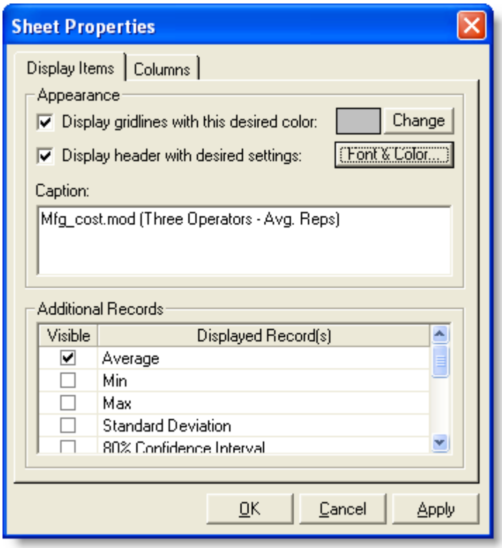
- **Report** Spreadsheets of the data collected during the simulation. see "Creating Reports" on page 382.
- **Category Charts** Bar charts of your data. see "Category Chart" on page 391.
- **State Charts** Stacked bar charts of state variables such as location and resource states. see "State Chart" on page 392.
- **Histogram** Displays graphically the frequency of occurrences of time plot data. see "Histogram" on page 394.
- **Time Plot** Shows variables, states and events over time. see "Selection Window" on page 396.

Sheet/Chart Properties

The title of the last option in the View menu will vary between Sheet Properties and Chart Properties, depending on whether a spreadsheet report or graphical chart is currently displayed. This selection brings up a window where you can choose the data you want to appear in the open report or chart.

Sheet Properties - Display Items

If you choose Sheet Properties while you are working with a Report, the following window appears. The default Display Items Tab is described below.



Appearance

☑ Display gridlines with this desired color:

☑ Display header with desired settings:

Caption:

Mfg_cost.mod (Three Operators - Avg. Reps)

Additional Records

Visible	Displayed Record(s)
☑	Average
□	Min
□	Max
□	Standard Deviation
□	80% Confidence Interval

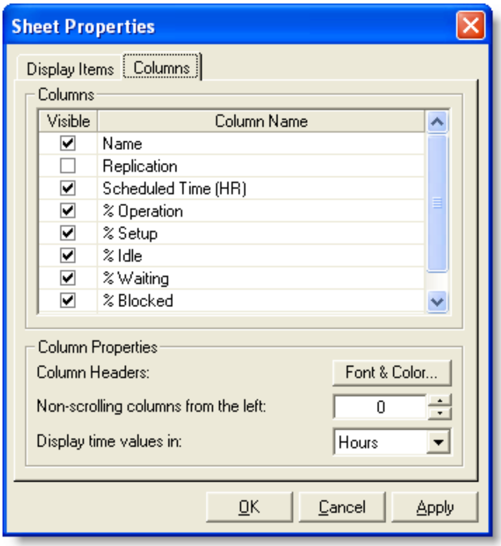
- **Display gridlines with this desired color:** When this box is checked gridlines will appear in the data sheet in the selected color. Unchecking this box will hide all grid lines for the data sheet.
- **Display header with desired settings:** When this box is checked the header will appear with the font styles selected when you click the Font & Color button. If you uncheck this box, no heading will be displayed above the sheet.
- **Caption:** The text displayed in the sheet heading. The default caption uses the following construct: [model name] ([scenario name] - [replication information]).
- **Additional Records:** The options in this area allow you to display additional replication information, which includes: the min and max, standard deviation, and various confidence interval information for the replications.

Please note

If you are working with a database that does not contain multiple replications, the Additional Records field of the Sheet Properties window is not applicable, and therefore not accessible.

Sheet Properties - Columns

The “Columns” tab for Sheet properties is described below.



Columns

Visible	Column Name
☑	Name
□	Replication
☑	Scheduled Time (HR)
☑	% Operation
☑	% Setup
☑	% Idle
☑	% Waiting
☑	% Blocked

Column Properties

Column Headers:

Non-scrolling columns from the left: 0

Display time values in: Hours

- **Columns:** You may choose to show or hide columns by checking or unchecking the boxes in this list. The available columns in this list will vary depending on the sheet you have selected.

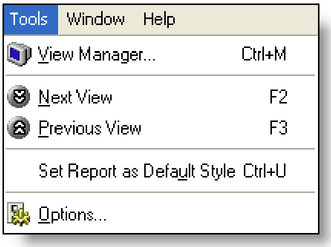
- **Column Headers:** Choose the font styles of the column headers.
- **Non-scrolling columns from the left:** Use this column to lock columns in place as you scroll horizontally through your sheet data. For example, entering “1” in this field will lock the left-most column in place, so that it will not scroll with the rest of the sheet’s columns.
- **Display time values in:** By default, time values are shown is hours, but you may choose to have time units shown in seconds, minutes, hours, days, or weeks.

Chart Properties

The Chart Properties dialog will vary depending on what type of chart you choose to view properties for. See each chart type description (beginning with “Category Chart” on page 391) for more information on the options available from its Chart Properties dialog.

Tools Menu

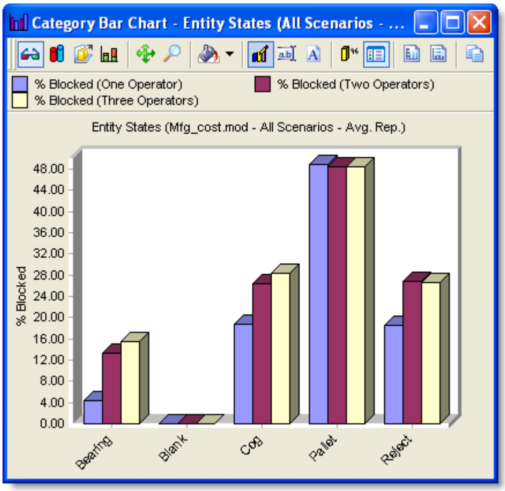
The Tools menu allows access to the View Manager and Output Viewer 3DR options.



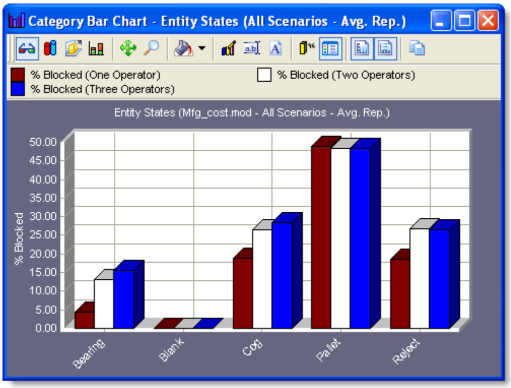
View Manager

The View Manager allows you to save customized report and chart views for quick retrieval. The View Manager is great for creating custom presentations. Views are associated with the model, so the next time the model is run the same view appears.

A view describes the way a report or chart appears. For instance, you may choose to create a category chart as explained on page 391.

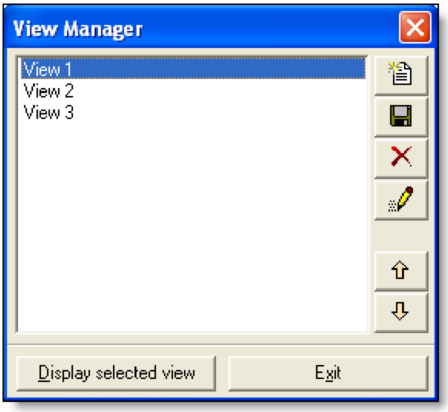

Category Bar Chart - Entity States (All Scenarios - ...
% Blocked (One Operator)
% Blocked (Two Operators)
% Blocked (Three Operators)
Entity States (Mfg_cost.mod - All Scenarios - Avg. Rep.)
% Blocked
48.00
44.00
40.00
36.00
32.00
28.00
24.00
20.00
16.00
12.00
8.00
4.00
0.00
Bearing
Blank
Cog
Pallet
Reject

That chart may then be customized using the tools in the Chart toolbar, page 389, and the right-click menu, page 379.


Category Bar Chart - Entity States (All Scenarios - Avg. Rep.)
% Blocked (One Operator)
% Blocked (Two Operators)
% Blocked (Three Operators)
Entity States (Mfg_cost.mod - All Scenarios - Avg. Rep.)
% Blocked
50.00
45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
Bearing
Blank
Cog
Pallet
Reject

In the example shown, the fonts, coloring and window size were changed, but the data remains the same.

This particular customized view of the chart can be saved using the View Manager.



The View Manager contains a field with the names of your views and the following tools:

- 📄 : Creates and saves a new view based on the report or chart you currently have selected in the Output Viewer 3DR.
- 💾 : Saves a view after you have modified it.
- ✗ : Deletes the view you have selected in the View Manager.
- ✎ : Renames a selected view.
- ⇧ , ⇩ : Moves the name of a selected view up or down within the list in the View Manager. This is helpful for creating an ordered list of customized views. The first view in the list is the initial view displayed after rerunning the model again and opening the Output Viewer 3DR.

How to Save a Customized View

1. Open a report or chart window. See "Reports and Charts" on page 374 for a list of possible reports and charts.

2. Customize the report or chart as desired using the tools in the Chart toolbar, page 389, and the right-click menu, page 379.

3. Open the Views Manager by clicking on the ▣ tool in the Output Viewer 3DR toolbar, or selecting View Manager from the Tools menu.

4. Click the New Icon, 📄 , in the View Manager and give the view a name.

5. Click OK to save your new view.

Next/Previous View

The Next/Previous View options in the Tools menu allows you to scroll through the views you have saved in the View Manager.

Set as Default Style

This option allows you to set the characteristics of a report or chart as your default style. This is helpful, for instance, if you customize the color scheme of a chart, and want it to apply to all the charts you open.

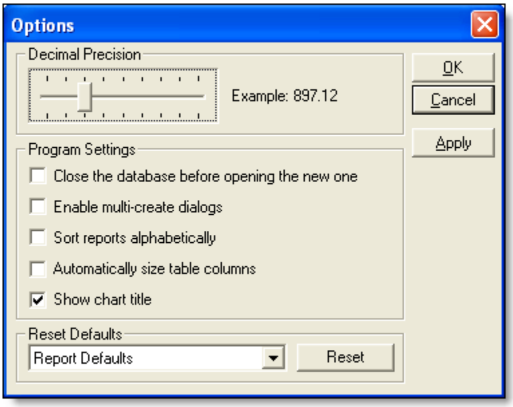
Setting a default style is different from saving a view, which will only save color and font changes for just one report or graph.

If a report or chart is selected when you choose the Set as Default Style option, all of the font and color changes you have made to the selected report or chart will be applied to any new reports or charts you open.

The default style settings can be restored from the Options window described next.

Options

The Options window allows you to change basic settings for the Output Viewer 3DR.



Options

Decimal Precision

Example: 897.12

Program Settings

☐ Close the database before opening the new one

☐ Enable multi-create dialogs

☐ Sort reports alphabetically

☐ Automatically size table columns

☑ Show chart title

Reset Defaults

Report Defaults

The decimal precision can be changed to increase or decrease the number of digits that appear for all real numbers that are displayed.

You can choose to have the current database close whenever you open a new database.

Selecting the check box for Enable Multi-create Dialogs will leave the Select Report and Select Chart windows open until you close them. This allows you to open multiple reports or charts at the same time without having to reopen the creation window each time.

This option is checked by default. If you want the selection widow to close when you open a report or chart, uncheck the box.

If you have changed your report or chart style as described in the previous heading, you can restore the default styles by using the Reset Defaults menu and Reset button.

How to Reset Default Styles

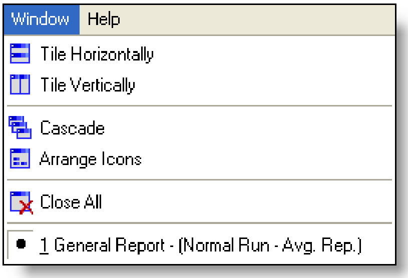
1. Select Report or Chart Defaults from the pull-down menu in the Options window.

2. Click the Reset button.

3. Click OK to close the Options window. The default styles will be reset for reports or charts.

Window Menu

The Window menu contains useful tools for making multiple data windows easier to view and compare with one another.



- **Tile Horizontally** Sizes and places each open window horizontally across the screen so none of them overlap. Since most data in the Output Viewer is arranged horizontally within a window, this option is very helpful for comparing data from one window with another.
- **Tile Vertically** Sizes and places each open window vertically across the screen. This makes comparing data across two or more windows convenient when the data in the window is arranged vertically.
- **Cascade** Places windows on top of each other in an overlapping manner.
- **Arrange Icons** If you have report or chart windows minimized within the Output Viewer 3DR window, this options will align them in a row along the bottom of the Output Viewer 3DR window.
- **Close All** Closes all open windows in the Output Viewer 3DR.
- **Open Windows** This last field in the Window menu displays the names of all the open windows. This is helpful for navigating

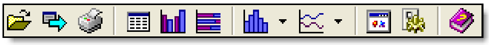
through data windows when you have many open at one time.

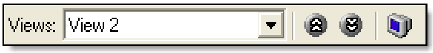
Help Menu

The Help menu gives you access to the Output Viewer's help file as well as information about the Output Viewer 3DR.

Toolbar

The Output Viewer 3DR Toolbar gives you quick access to some of the options found in the Menu bar.





File Tools

The File tools allow you to quickly open data files, export data, and print files.

- : Opens an .idb or .rdb data file.
- : Exports data.
- : Prints the data.

For more information on these features see "File Menu" on page 374.

Report and Chart Tools

The Report and Chart tools gives quick access to options found in the View menu.

- : Creates a report.
- : Creates a category chart.
- : Creates a state chart.
- : Creates a histogram.
- : Creates a time plot chart.

For more information on these features see "View Menu" on page 374.

Option Tools

These tools allow you to access some of the features in the Tools menu as well as the Sheet Properties window.

- : Displays Sheet Properties.
- : Displays the Options window.

For more information of Sheet Properties, see "Sheet/Chart Properties" on page 374. For more information on the Options windows, see "Tools Menu" on page 376.

Help Tool

- : Starts the Output Viewer 3DR Online Help.

View Manager Tools

The View Manager tools are located on the far right of the toolbar. These tools allow you to quickly select a view.

-

: Allows you to select a view from the pull-down menu.
- : Scrolls through your views.
- : Displays the View Manager window.

For more information on Views and the View Manager, see "View Manager" on page 376

Right-click Menu

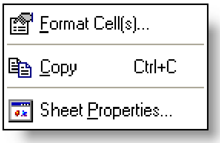
The Right-click menu provides a shortcut for editing the appearance and structure of your reports and charts.

The Right-click menu is available in report and chart windows. Additionally, the options that appear when you right-click in the Output Viewer

3DR vary depending on which area you right-click on.

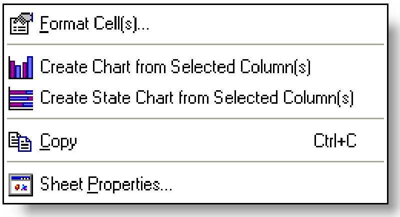
Right-click Menu in Reports

When you right-click anywhere in a report window, the following menu appears.



- **Format Cells** This option brings up a window that allows you to edit the size and alignment of the cell fonts, and the color of the cells.
- **Copy** Copy transfers the contents of the highlighted cells to the clipboard. This is useful for transferring select information to a separate spreadsheet.
- **Sheet Properties** This option brings up the Sheet Properties of the current report. For more information on this window see "Sheet/Chart Properties" on page 374.

When you right-click anywhere in a report window that contains numerical values, the Right-click menu contains two additional options depending on whether the values are percentage values or not.


Format Cell(s)...
Create Chart from Selected Column(s)
Create State Chart from Selected Column(s)
Copy Ctrl+C
Sheet Properties...

- Create Chart If you right-click on a column of numerical value, including percentage value, you can choose to have that data displayed as a Category Chart. The Category Chart that's created is the same as one created using the Category Chart option. See "Category Chart" on page 391 for more information.
- Create State Chart If you right-click on a column that contains percentage values, you can choose to have that data displayed, together with all other percentage columns on the sheet, as a State Chart. The State Chart that's created is the same as one created using the State Chart option. See "State Chart" on page 392 for more information.

Right-click Menu in Charts

The Right-click menu that appears for charts is different than the menu for reports.

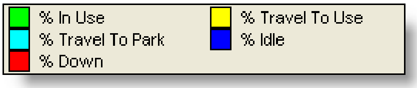
There are three areas within a Chart window that you can right-click to bring up a different menu:

- On the legend area.
- In the Chart area.
- On a title or label.

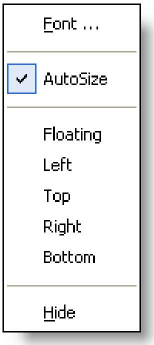
Right-click Menu for the Legend Area

The legend area is located within the Chart window, and gives an explanation of the chart's color codes.

An example of a legend area is shown below.



When you right-click anywhere in the legend area, the following menu appears.

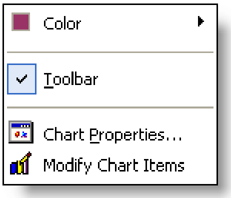


There are four areas in this menu:

- **Font** This features lets you change the font options for the legend labels.
- **AutoSize** The AutoSize option resizes the legend area to fit the labels in the legend.
- **Legend Position** The position of the legend area can be moved to any one of the four sides of the Chart window, or it can float in its own window.
- **Hide** Selecting this option makes the legend disappear. You can bring the legend back by opening the Chart Properties window in the View menu.

Right-click Menu for the Chart Area

Right clicking anywhere in the Chart window, except for the legend area, brings up the following window.

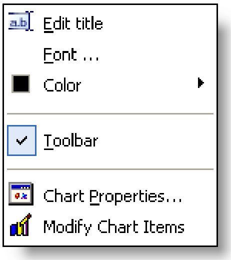


- **Color** This will allow you to change the color of the area your mouse was positioned over when you opened the Right-click menu. This could be the color of the background or bars in the chart.
- **Toolbar** Toggles the toolbar for chart windows on and off. This is helpful, for example, to make the chart appear less cluttered during a presentation of the data or on a printout of the chart. For more information on the chart window toolbar see “Chart Toolbar” on page 389.
- **Chart Properties** Opens the Properties window. For more information on this window see “Sheet/Chart Properties” on page 374.
- **Modify Chart Items** This options opens the chart’s data dialog, which was initial dialog shown when you first created the chart. This option is only available for Histogram and Time Plot chart types.

  For more information on the data dialog see the corresponding chart descriptions, beginning with “Category Chart” on page 391.

Right-click Menu for Titles and Labels

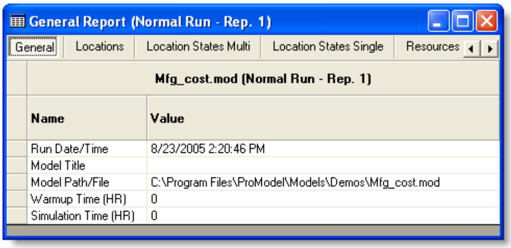
Titles and labels are the text within the chart area. Right-clicking on either of these two will bring up a Right-click menu similar to that of a chart area’s menu, but with two additional options.



- **Edit Title** Edits the text for the title of the chart or axes.
- **Font** Allows you to change the font characteristics for the titles and labels.

Creating Reports

Reports contain the numerical data that is collected during your simulation.



Mfg_cost.mod (Normal Run - Rep. 1)

Name	Value
Run Date/Time	8/23/2005 2:20:46 PM
Model Title	
Model Path/File	C:\Program Files\ProModel\Models\Demos\Mfg_cost.mod
Warmup Time (HR)	0
Simulation Time (HR)	0

This information is presented in spreadsheet form, and grouped into the following categories:

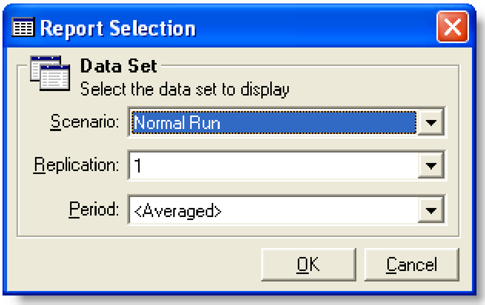
- General
- Entity Activity
- Entity Costing
- Entity States
- Failed Arrivals
- Location Costing
- Location States Multi
- Location States Single/Tank
- Locations
- Logs
- Node Entries
- Resource Costing
- Resource States
- Resources
- Variables

For more information on each of these categories see “Report Data” on page 383.

How to create a report

1. Choose the Reports option from the View menu or the Reports icon, ▦ , from the Toolbar.

2. If the model you ran had multiple scenarios or replications, the following window appears.



Data Set
Select the data set to display

Scenario: Normal Run

Replication: 1

Period: <Averaged>

3. From this window you may choose which scenarios and replications to view. If you chose to run the simulation with Batch Mean or Periodic reporting, you may also choose a Period.

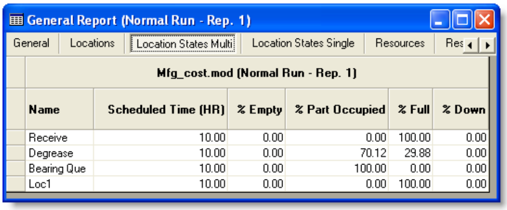
4. Click on the Create button. This will open a new window with the chosen report.

Please note

If your model was run for only one replication and one scenario, the report window will open immediately after selecting the Reports option.

Report Window

The Report window contains the categorized information from your simulation.



Mfg_cost.mod (Normal Run - Rep. 1)

Name	Scheduled Time (HR)	% Empty	% Part Occupied	% Full	% Down
Receive	10.00	0.00	0.00	100.00	0.00
Degrease	10.00	0.00	70.12	29.88	0.00
Bearing Que	10.00	0.00	100.00	0.00	0.00
Loc1	10.00	0.00	0.00	100.00	0.00

The data appears in spreadsheet format, and is grouped by the categories listed by tabs along the top of the Report window.

Selecting a tab will bring up a new table, and additional tabs can be accessed using the left and right arrows, ◀▶.

The caption of a report can be renamed by either double clicking on it and then typing in the new name or opening the sheet properties window and typing a new name in the Caption area.

Report Data

The spreadsheet for each button contains the following information.

Please note

The default time units shown in the report will be one unit longer than the unit selected in the Simulation Options dialog box. For example, if you select minutes in Simulation Options, the time displayed in the report will be in Hours. You may also notice that Day is the last option available in Simulation Options, in which case the time unit will be in weeks for the report.

General

- **Run Date/Time** The date and time the model was run for the displayed output.
- **Model Title** The name of the model.
- **Model Path/File** The path and name of the model file.
- **Warmup Time** The amount of initial time the simulation spent in a warmup state. No statistics were gathered during this time. The time unit used is one unit longer than defined in the Clock Precision field of the Simulation Options dialog.
- **Simulation Time** The total simulation time, including the warmup time. The time unit used is one unit longer than defined in the Clock Precision field of the Simulation Options dialog.

Entity Activity

Entity activity is reported for only those entities that have exited the system.

- **Total Exits** The number of entities that completely exit the system either through the EXIT routing or when they are joined, renamed, or combined. In some cases, entities also exit the system when you use the SPLIT AS, UNGROUP, or ROUTE statements.
- **Current Quantity In System** The total number of entities remaining in the system at the time the simulation ends. These are entities that have not exited.
- **Average Time In System** The average total time the entity spends in the system.
- **Average Time In Move Logic** The average time the entity spent traveling between locations, including any delays incurred in move logic.
- **Average Time Wait For Res** The average time the entity spent waiting for a resource or another entity (to join or combine). Also includes time waiting in queue behind a blocked entity.
- **Average Time In Operation** The average time the entity spent processing (i.e., WAIT or USE statements) at a location or traveling on a conveyor/queue.
- **Average Time Blocked** The average time the entity spent waiting for a destination location to have available capacity.

Entity Costing

- **Explicit Exits** The number of entities that have explicitly exited. Whenever an entity exits the system, it is an explicit exit except in the following cases:

- When an entity JOINS or COMBINES with another entity, it implicitly exits the system, and is reported as an exit in the Entity Activity report. However, for costing purposes, the entity did not explicitly exit, but its costing information was added to the entity it was JOINED or COMBINED with.
- When an entity LOADS or GROUPS with another entity, and the entire LOADED or GROUPED entity exits the system, the original entity implicitly exits the system, and is reported as an exit in the Entity Activity report. However, for costing purposes, the original entity did not explicitly exit, but its costing information was added to the entire load or group.

- **Total Cost Dollars** 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** % 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.

Please 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.

Entity States

Reported by entity type for only those entities that have exited the system.

- **% In Move Logic** The percentage of time the entity spent traveling between locations, including any delay time incurred in move logic.
- **% Waiting** The percentage of time the entity spent waiting for a resource, a WAIT UNTIL condition, another entity to join or combine, or behind other entities. (100% - Sum of %'s for all other states.)
- **% In Operation** The percentage of time the entity spent in processing at a location or traveling on a conveyor/queue. If an entity is on a conveyor behind another entity which is blocked because the next location is unavailable, the time the entity spent behind the other entity is considered % in Operation.
- **% Blocked** The percentage of time the entity spent waiting for a the next location to become available.

Failed Arrivals

The number of entities that failed to arrive at a specific location due to insufficient capacity.

Locations Costing

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

Please 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 Dollars** Total Cost = (Operation Cost + Resource Cost)
- **% Total Cost** Refers to location's percentage of the sum of all location costs

Location Setup

- **Name** The name of the location where the setup downtime occurred.
- **Entity** The name of the entity that caused the setup downtime.
- **Total Setups** The number of times this location/entity combination resulted in a setup downtime.
- **Avg Time Per Setup (HR)** The average time in hours the location was down for each setup downtime.

Location States (Multiple Capacity)

- **Scheduled Time** The total amount of time the location was scheduled to be available. This value is now in decimal format, not truncated. (Excludes off-shift time, break time, and scheduled downtimes.)
- **% Empty** The percentage of time the location had no entities.
- **% Partially Occupied** The percentage of time the location has entities but was not filled to capacity (100% of time - %Full - %Empty).
- **% Full** The percentage of time the location was full to capacity with entities.
- **% Down** The percentage of time the location was down as the result of unscheduled downtimes. This does not exclude the possibility of overlap with any of the previous three states.

Location States (Single Capacity/ Tank)

- **Scheduled Time** The total amount of time the location was scheduled to be available. This value is now in decimal format, not truncated. (Excludes off-shift time, break time, and scheduled downtimes.)
- **% Operation** The percentage of time the location was actually processing an entity.
- **% Setup** The percentage of time the location spent in setup in order to process the entities.
- **% Idle** The percentage of time no entities were at the location, but the location was not down.
- **% Waiting** The percentage of time the location was waiting for a resource, another entity, or a WAIT UNTIL condition in order to begin processing or move to the next location. Any delays in processing move logic (even WAIT statements) are counted as waiting time and include the following statements:
  - ACCUM
  - COMBINE
  - GROUP
  - JOIN
  - LOAD
  - MATCH
- **% Blocked** The percentage of time entities spent waiting for a freed destination.
- **% Down** The percentage of time the location was down due to unscheduled downtimes.

Locations

- **Scheduled Time** The total amount of time the location was scheduled to be available.

This value is in decimal format, not truncated. (Excludes off-shift time, break time, and scheduled downtimes.)

- **Capacity** The capacity defined in the Locations module for this location.
- **Total Entries** The total number of entities that entered the location, not including entities arriving to be joined and loaded. Entities split, unloaded, or ungrouped from another entity at a location do not count as additional entries. Arriving entities that have been previously grouped or loaded to form a single entity only count as one entry.
- **Average Time Per Entry** The average time each entry spent at the location. This time may include partial times from the beginning and end of the actual run time.
- **Average Contents** The average number of entries at the location.
- **Maximum Contents** The maximum number of entries which occupied the location over the course of the simulation.
- **Current Contents** The number of entities remaining at the location when the simulation ended.
- **% Utilization** The percentage of capacity occupied, on average, during the simulation.

$$\frac{\text{Cumulative Occupancy Time x 100}}{\text{Capacity x Scheduled Time}}$$

Cumulative Occupancy Time refers to the sum of the clock time each entity spends at a location for processing.

Logs

- **Numbers of Observations** The number of log entries that occurred during the simulation for the given Log Name.
- **Minimum Value** The minimum log entry value during the simulation for the given Log Name.
- **Maximum Value** The maximum log entry value during the simulation for the given Log Name.
- **Average Value** The average value of all log entries during the simulation for the given Log Name.

Please note

Log statistics are set up by the user with a LOG statement to track the time entities spend between any two points in the model.

Node Entries

A node entry summary is generated for each nonpassing path network in the system. It contains the following information.

- **Path Name** The name of the path the node resides on.
- **Total Entries** The number of times that a resource entered the path node.
- **Blocked Entries** The number of times a resource tried to claim a path node occupied by another resource.

Resources Costing

- **NonUse Cost Dollars** NonUse Cost = (1-% Utilization) * Scheduled Time * Rate
- **% NonUse Cost** Refers to the resource's percentage of the sum of all nonuse costs
- **Usage Cost Dollars** 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 Dollars** Total Cost = Usage Cost + NonUse Cost

- **% Total Cost** Refers to the resource's percentage of the sum of all resource costs

Resource States

- **Scheduled Time** The total amount of time the resource was scheduled to be available. (Excludes off-shift time, break time, and scheduled downtimes.)
- **% In Use** The percentage of time the resource spent transporting or processing an entity, or servicing a location or other resource that was down. This also includes deposit time.
- **% Travel To Use** The percentage of time the resource spent traveling to a location or other resource to transport or process an entity, or to service a location or other resource. This also includes pickup time. (This information is not available if your model does not have dynamic resources.)
- **% Travel To Park** The percentage of time the resource spent traveling to a path node to park or traveling to its downtime node. (This information is not available if your model does not have dynamic resources.)
- **% Idle** The percentage of time the resource was available but not in use.
- **% Down** The percentage of time the resource was unavailable due to unscheduled downtimes.

Resources

- **Units** The number of units defined in the Resources module for that resource.
- **Scheduled Time** The total amount of time the resource was scheduled to be available. (Excludes off-shift time, break time, and scheduled downtimes.)
- **Number of Times Used** The total number of times the resource has been acquired to transport or process an entity or to service locations or other downed resources.
- **Average Time Per Usage** The average time the resource spent transporting or processing an entity, or servicing a location or other resource. Includes any pickup and drop-off time as well as any blocked time while in use. (See note on time units.)
- **Average Time Travel To Use** The average time the resource spent traveling to a location or other resource to transport or process an entity, or to service a location or other resource. Does not include any pickup time, but does include any blocked time. (This information is not available if your model does not have dynamic resources.)
- **Average Time Travel To Park** The average time the resource spent traveling to either a park node or a downtime node. (This information is not available if your model does not have dynamic resources.)
- **% Blocked In Travel** The percentage of time the resource was unable to move to a destination because the next path node along the route of travel was blocked (occupied by another resource). (This information is not available if your model does not have dynamic resources.)
- **% Utilization** The percentage of time the resource spent traveling to be used, transporting or processing an entity, or servicing a location or other resource.

Total Travel to Use Time + Total Time In Usage
x 100
Total Scheduled Time

Please note

ProModel reports resource groups and multi-unit resources both by unit and collectively. The collective unit (aggregate) report for a resource totals the first three fields and averages the last five fields discussed above.

Variables

- **Total Changes** The total number of times the value of the given variable changed during the simulation.
- **Average Time Per Change** The average time a given variable remained at any one value.
- **Minimum Value** The lowest value of the variable during the simulation.
- **Maximum Value** The highest value of the variable during the simulation.
- **Current Value** The final value of the variable when the simulation ended.
- **Average Value** The average value of the variable during the simulation. This value is time weighted.

Please note

In the report, if a variable name is followed by an asterisk (*), the variable is an observation-based variable. Otherwise, it is a time-weighted variable. This determination is made in the Variable edit table where the variable is defined.

The initial value for observation-based variables is not accounted for in the statistics.

Creating Charts

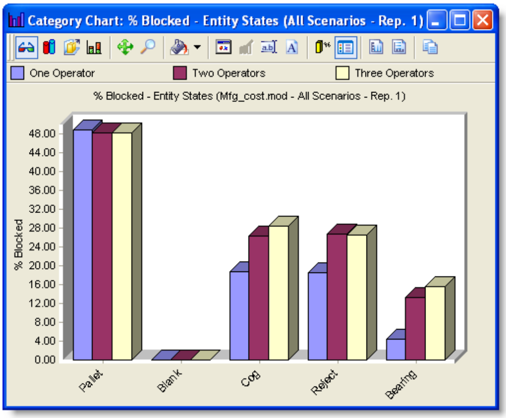
Charts provide visual representations of the data contained in Reports.

The Output Viewer 3DR can display four main types of charts with several sub-types:

- Category Charts
- State Charts
- Histograms
  - Time-Weighted Values
  - Simple Values
- Time Plots
  - Time-Weighted Values
  - Simple Values
  - State Values
  - Counts

Chart Window

The chart window contains your chart and options to modify your chart.

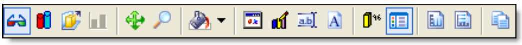


Regardless of the type of chart you open, all chart windows have options and fields in common:

- Chart Toolbar
- Legend Field
- Chart Area
- Display and Alias

Chart Toolbar

The toolbar that appears in a Chart window gives you several options to change the way the information is presented in your chart.



- : Toggles between a three-dimensional and two-dimensional bars in your chart. Three-dimensional is the default view.
- : Toggles between bars and cylinders in the Chart area.
- : Expands the chart along the z-axis to show multiple scenarios in layers. This option will not be available if only one scenario was run.
- : Shows the results in a stacked bar chart.
- : When the chart is three-dimensional, this option will allow you to rotate the view using the cursor.
- : Use this tool to enlarge regions of your chart. Select this tool then click and drag to outline the chart area you want to enlarge. You can continue to zoom in on an area, or you can reset the view to the default zoom level by selecting this tool again.
- : Use this tool to change the colors in the Chart area. Selecting this tool will open a menu with color options. Choosing a color will change your cursor to a paint bucket. Position the bucket over the area you want to change, and click to fill the area with the new color.
- : Opens the chart's properties dialog.

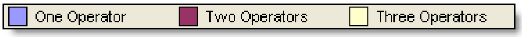
- : Opens a dialog, which allows you to modify the items displayed in the chart. This option is only available for Histogram and Time Plot charts.
- : Allows you to change the chart's titles and labels.
- : Allows you to change the font characteristic of the chart's titles and labels.
- : Toggles the point labels on and off. Point labels are the numbers that the bars in your chart represent. These numbers will appear at the top of the bars when this option is selected.
- : Toggles the point makers on and off. Point markers are small graphics that show the data points on your graphs.
- : Toggles the Legend area on and off.
- : Toggles the Display and Alias field on and off. For more information on this field see "Display and Alias" on page 390.
- : Toggles the vertical grid lines in the chart on and off.
- : Toggles the horizontal grid lines in the chart on and off.
- : Copies the chart to the Windows clipboard as a bitmap, a metafile, or as plain text (data only).

Please note

The Bar/Cylinder, , tool is not available for Simple Values and Time-weighted Values time plot charts, since they do not use bars to represent data.

Legend Field

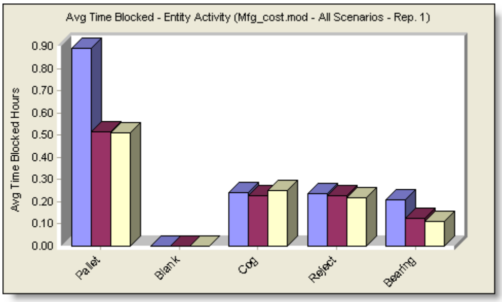
The Legend field shows the definition and color-coding of the bars in your chart.



🟦 One Operator 🟥 Two Operators 🟨 Three Operators

Chart Area

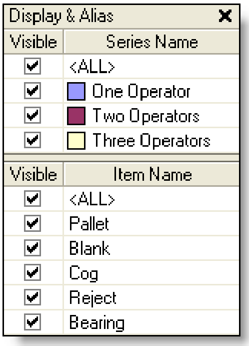
The Chart area contains your chart.



The appearance of this chart will depend on the type of chart you are viewing; i.e. category, histogram, etc.

Display and Alias

The Display and Alias field, which can be toggled on and off from the toolbar, allows you to quickly select the data items you wish to display in the chart and optionally assign aliases to the data items.



Display & Alias ✕

Visible	Series Name
☑	<ALL>
☑	🟦 One Operator
☑	🟥 Two Operators
☑	🟨 Three Operators

Visible	Item Name
☑	<ALL>
☑	Pallet
☑	Blank
☑	Cog
☑	Reject
☑	Bearing

Select which series and items you would like displayed in the chart by checking the box in the

Visible column. To show or hide all the items you can click check or uncheck the box in the <ALL> row or click on any box while holding the CTRL key.

Double click on a series or item name to rename it with an alias.

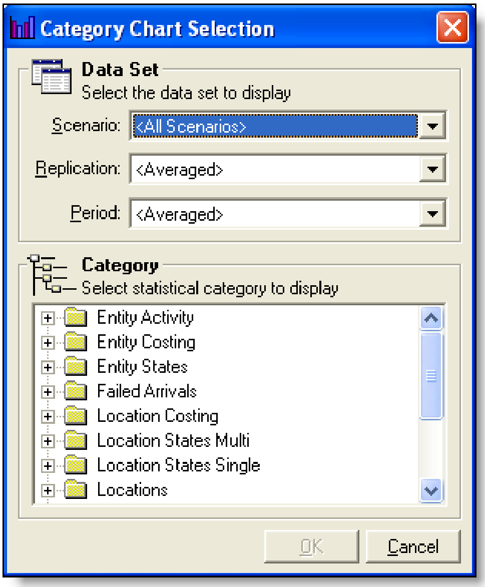
Category Chart

Category Charts displays bar charts for columns of data found in reports.

The Category Chart is an excellent way to visually compare the different items in your reports. It is also helpful for organizing data attractively for presentations.

To display a Category Chart, select Category Chart from the View menu or the Category Chart icon from the Toolbar. As a shortcut, you can also right-click on any column of data in a report and select Category Chart from the drop-down menu.

When you choose to view a Category Chart, the Category Chart Selection dialog will be displayed.



Category Chart Selection

Data Set

Select the data set to display

Scenario: <All Scenarios>

Replication: <Averaged>

Period: <Averaged>

Category

Select statistical category to display

From this menu you can select the scenarios, replications, periods (only available if Batch Mean or Periodic Output Reporting is selected in the Simulation Options dialog) and categories of the data you want displayed.

For information on each category, see “Report Data” on page 383.

Category Chart Properties

Choosing the properties option for a category chart will open the Chart Properties dialog.

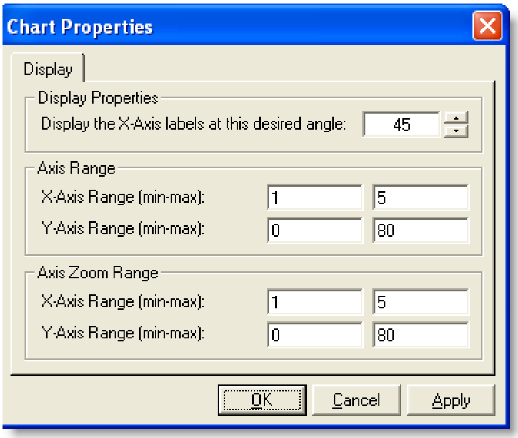


Chart Properties

Display

Display Properties

Display the X-Axis labels at this desired angle: 45

Axis Range

X-Axis Range (min-max): 1 5

Y-Axis Range (min-max): 0 80

Axis Zoom Range

X-Axis Range (min-max): 1 5

Y-Axis Range (min-max): 0 80

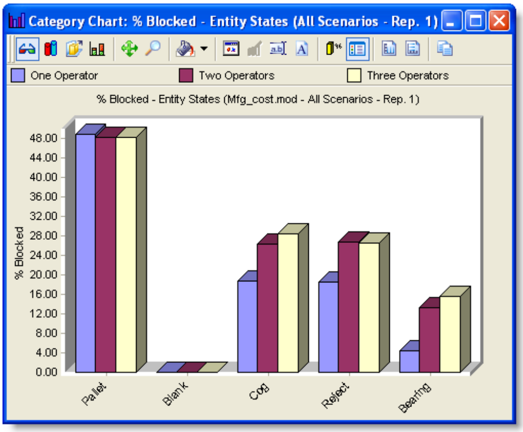
- **Display Properties:** Select the angle of the text for the X-Axis labels.
- **Axis Range - X-Axis Range (min-max):** These two boxes show the range of items displayed in the X-Axis.
- **Axis Range - Y-Axis Range (min-max):** In the first box choose the value for the minimum range for the Y-Axis. The maximum range is entered in the second box.
- **Axis Zoom Range - X-Axis Range (min-max):** Choose which range of X-Axis items to zoom in on by using these two boxes.
- **Axis Zoom Range - Y-Axis Range (min-max):** Choose which range of Y-Axis values to zoom in on by using these two boxes.

Please Note

Changes to a chart’s properties does not apply to other charts or persist when the chart is closed and then regenerated.

Category Chart Example

An example of a Category Chart is shown below.



Please Note

Positioning your cursor over a bar on the chart will display specific information about that particular bar. The appearance and structure of your chart can be edited using the chart toolbar or the right-click menu. For more information see “Chart Toolbar” on page 389 or “Right-click Menu” on page 379.

State Chart

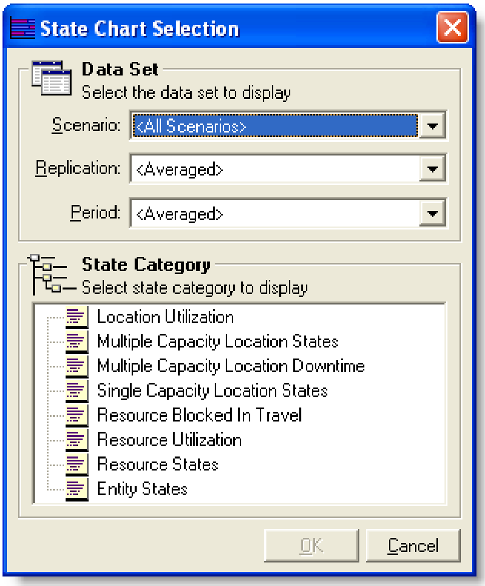
State Charts show stacked bar charts of location, resource, and entity states. States describe the condition of these objects.

This type of chart is useful for quickly seeing what your locations, resources, and entities spent their time doing. Such visualizations can help identify problem areas in your process.

To display a State Chart, select State Chart from the View menu or the State Chart icon from the Toolbar. As a shortcut, you can also right-click on any column of data in a report containing percentage values and select State Chart from the drop-down menu.

When you choose to view a State Chart, the State Chart Selection dialog will be displayed.

State Chart Selection Window



State Chart Selection

Data Set
Select the data set to display

Scenario: <All Scenarios>

Replication: <Averaged>

Period: <Averaged>

State Category
Select state category to display

- Location Utilization
- Multiple Capacity Location States
- Multiple Capacity Location Downtime
- Single Capacity Location States
- Resource Blocked In Travel
- Resource Utilization
- Resource States
- Entity States

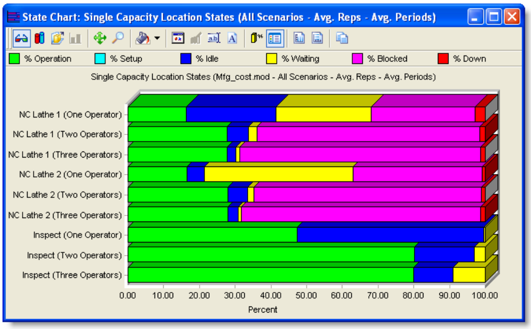
From this menu you can select the scenarios, replications, periods (only available if Batch Mean or Periodic Output Reporting is selected in the Simulation Options dialog) and state categories of the data you want displayed.

State Chart Properties

The State Chart Properties window is functionally equivalent as the Category Chart Properties window. See "Category Chart Properties" on page 392.

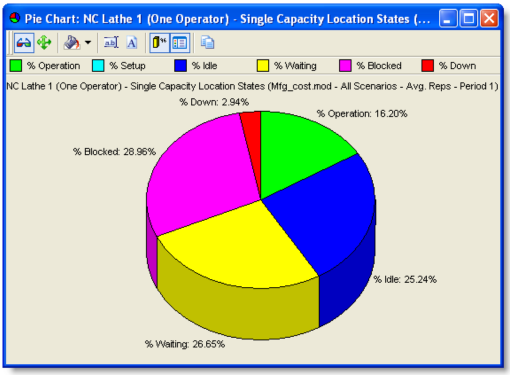
State Chart Example

An example of a State Chart is shown below.



This chart shows the percentage of time by scenario each location spent in the six different states: operation, setup, idle, waiting, blocked, and down.

Right click on any bar in the chart, and choose Create Pie Chart to have that item's state information displayed as a pie chart.



Please Note

Positioning your cursor over a bar on the chart will display specific information about that particular bar. The appearance and structure of *your chart can be edited using the chart toolbar or the right-click menu. For more information see "Right-click Menu" on page 379.*

Histogram

A histogram is a bar chart showing the percentage of time or times that time plot data fell within a particular range of values.

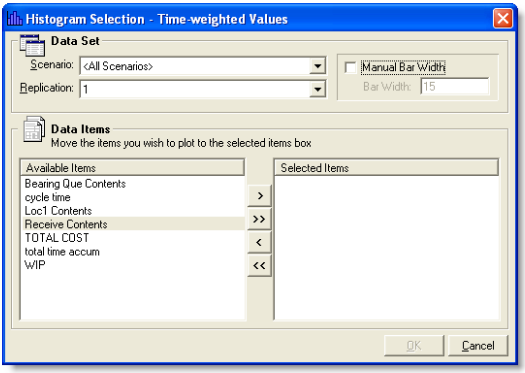
There are two types of histograms that can be displayed:

- **Time-weighted Values** Percentage of total simulation time that the values fell into a specific range. (e.g. the Contents variable of a location.)
- **Simple Values** Percentage of total simulation time that a variable was within a range of values (e.g. cycle times).

Time Weighted Values Histogram

The time weighted values histogram shows the percentage of total simulation time the values fell into specific ranges. The example further explains how values are grouped.

Selection Window



Histogram Selection - Time-weighted Values

Data Set

Scenario: <All Scenarios>

Replication: 1

□ Manual Bar Width

Bar Width: 15

Data Items

Move the items you wish to plot to the selected items box

Available Items: Bearing Que Contents, cycle time, Loc1 Contents, Receive Contents, TOTAL COST, total time accum, WIP

Selected Items:

Choose the scenarios and replications you would like displayed for the data item selected. Double click on an item to move it from the Available Items area to the Selected Items area or back again. You may also use the arrow buttons to move items between the two areas.

If you check the Manual Bar Width option, you may choose the range value for each bar shown in the chart. If you leave this field unchecked, 3DR will automatically calculate the range value for each bar, which will not necessarily be the value in the Bar Width field.

Chart Properties

Choosing the properties option for time weighted values histogram will open a window with two tabs: Data and Display

Chart Properties: Data

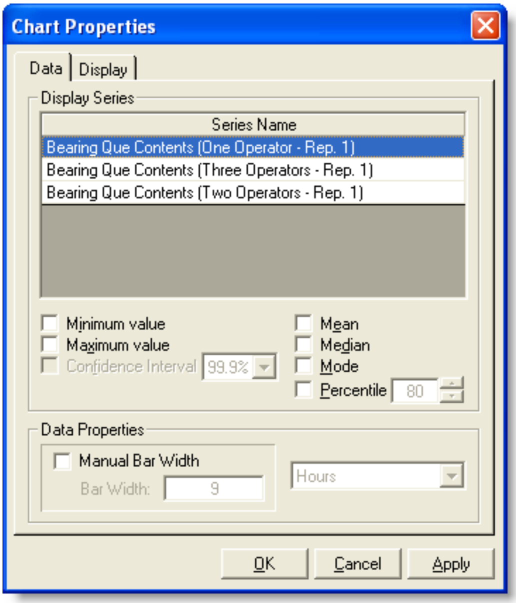


Chart Properties

Display Series

Series Name
Bearing Que Contents (One Operator - Rep. 1)
Bearing Que Contents (Three Operators - Rep. 1)
Bearing Que Contents (Two Operators - Rep. 1)

□ Minimum value

□ Maximum value

□ Confidence Interval 99.9%

□ Mean

□ Median

□ Mode

□ Percentile 80

Data Properties

□ Manual Bar Width

Bar Width: 9

Hours

- **Display Series:** This is a list of the series data displayed in the histogram.
- **Min/Max Values:** Choose whether to display the minimum and/or maximum values for all replications in the histogram.

- **Mean/Median/Mode Values:** Choose whether to display the mean, median, and/or mode values for all replications in the histogram.
- **Percentile:** Displays the maximum value the series could reach for the chosen percentage of total simulation time. For example, if you choose to display an 80 percentile for a location's contents, 80% of the simulation time the location's contents were at the displayed value or less.
- **Data Properties:** Displays the time units used to calculate the histogram's values, and optionally allows you to choose your own bar widths, which are the range of values each bar will display. Entering a smaller bar width results in more bars, while the greater the bar width the fewer bars will be displayed.

Chart Properties: Display

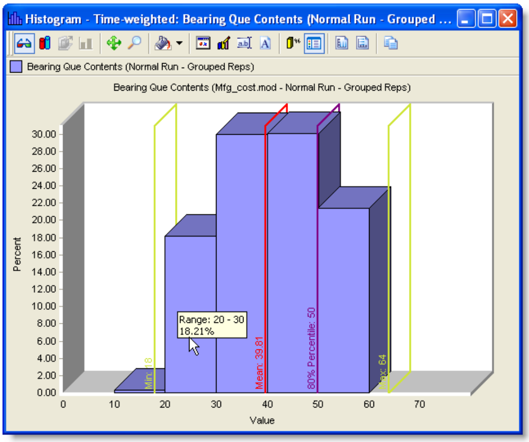
The Display tab for the time-weighted histogram properties window is functionally equivalent as the Category Chart Properties: Display window, with the exception of the "Discrete label the X-Axis" option.

- **Discrete label the X-Axis:** When this option is checked the full range of each bar is shown in the X-Axis.

See "Category Chart Properties" on page 392.

Chart Example

The example chart below shows the time-weighted histogram contents of a queue.


Histogram - Time-weighted: Bearing Que Contents (Normal Run - Grouped ...
Bearing Que Contents (Normal Run - Grouped Reps)
Bearing Que Contents (Mfg_cost.mod - Normal Run - Grouped Reps)
Percent
30.00
28.00
26.00
24.00
22.00
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Range: 20 - 30
18.21%
Min: 18
Mean: 39.81
80% Percentile: 50
Max: 64
0
10
20
30
40
50
60
70
Value

Along the X-Axis is the contents of the queue divided into bar units of 10. The Y-Axis shows the percentage of total simulation time the contents of the queue were at the values along the X-Axis.

The mouse over shows that for 18.21% of the total simulation time, for all replications, the contents of the queue were between exactly 20 and almost 30 (29.99).

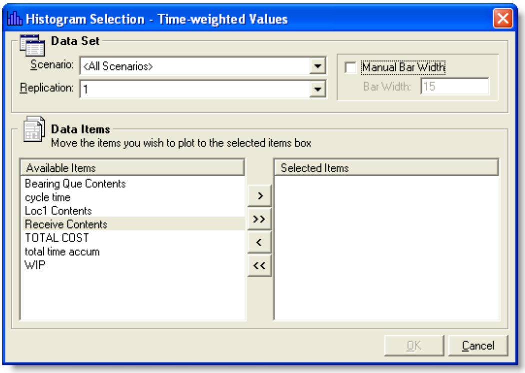
The average minimum and maximum contents of the queue, for all replications, were 18 and 64 respectively, while the mean contents were 39.81.

The 80% Percentile line shows that for 80% of the total simulation time the contents of the queue were 50 or less.

Simple Values Histogram

The simple values histogram shows when, by percentage of total simulation time, a variable's value was within a certain range.

Selection Window



Histogram Selection - Time-weighted Values

Data Set

Scenario: <All Scenarios>

Replication: 1

□ Manual Bar Width

Bar Width: 15

Data Items

Move the items you wish to plot to the selected items box

Available Items:
- Bearing Que Contents
- cycle time
- Loc1 Contents
- Receive Contents
- TOTAL COST
- total time accum
- WIP

Selected Items

Choose the scenarios and replications you would like displayed for the data item selected. Double click on an item to move it from the Available Items area to the Selected Items area or back again. You may also use the arrow buttons to move items between the two areas.

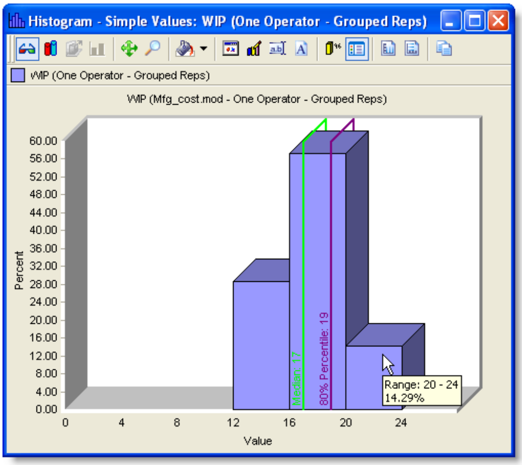
If you check the Manual Bar Width option, you may choose the range value for each bar shown in the chart. If you leave this field unchecked, 3DR will automatically calculate the range value for each bar, which will not necessarily be the value in the Bar Width field.

Chart Properties

The chart properties for the simple values histogram are the same as for the time-weighted values histogram. See "Chart Properties" on page 394.

Chart Example

The example chart below shows the simple values histogram for the variable WIP.



The X-Axis shows the value of the WIP variable divided into bar units of 4. The Y-Axis shows the percentage of times the WIP variable fell into the ranges shown on the X-Axis.

This chart also shows the median value of the WIP value for all replications. The 80% Percentile shows that for 80% of the counts the value the WIP variable was 19 or less.

Time Plot

Time plots show state and performance values as they occur over time. This allows you to see trends and spikes in activity as the simulation progressed.

Time plots can only be displayed if the user has selected Time Plot as the statistics type for objects as the model was being built.

There are four types of time-plot statistics that can be displayed:

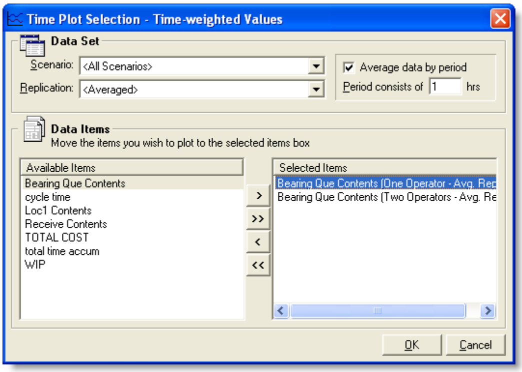
- **Time-weighted Values** Variable values weighted by time (e.g. the Contents variable of a location.)
- **Simple Values** Variable values as they change over time (e.g. cycle times).
- **State Values** Shows location states over time.
- **Counts** The number of occurrences of some particular event (e.g. the number of exits from a location).

Time Weighted Values Time plot

The time-weighted time plot chart shows variable values over time.

The chart's X-Axis shows time periods, while the Y-Axis shows variable values for each time period.

Selection Window



Time Plot Selection - Time-weighted Values

Data Set

Scenario: <All Scenarios>

Replication: <Averaged>

☑ Average data by period

Period consists of 1 hrs

Data Items

Move the items you wish to plot to the selected items box

Available Items	Selected Items
Bearing Que Contents	Bearing Que Contents (One Operator - Avg. Rep
cycle time	Bearing Que Contents (Two Operators - Avg. Re
Loc1 Contents	
Receive Contents	
TOTAL COST	
total time accum	
WIP	

Choose the scenarios and replications you would like displayed for the data items selected. Double click on an item to move it from the Available Items area to the Selected Items area or back again. You may also use the arrow buttons to move items between the two areas.

If you check the "Average data by period" box, all the values in each period of selected time will be averaged together. If left unchecked, each value will be displayed.

Chart Properties

Choosing the properties option for time weighted values time plot will open a window with two tabs: Data and Display

Chart Properties: Data

The chart properties:data for the time-weighted values time plot are the same as for the time-weighted values histogram. See "Chart Properties" on page 394.

Chart Properties: Display

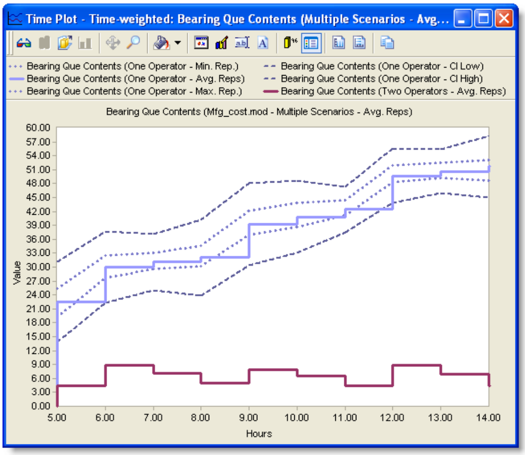
The Display tab for the time-weighted time plot properties window is functionally equivalent as the Category Chart Properties: Display window, with the exception of the "Line Properties" option.

- **Line Properties:** Choose the pattern and thickness for the lines on the chart.

See "Category Chart Properties" on page 392.

Chart Example

The example chart below compares the time-weighted time plot contents of a queue for two scenarios.



The X-Axis shows the time periods for the chart. The Y-Axis shows the average content values of the queue for the time periods in the X-Axis.

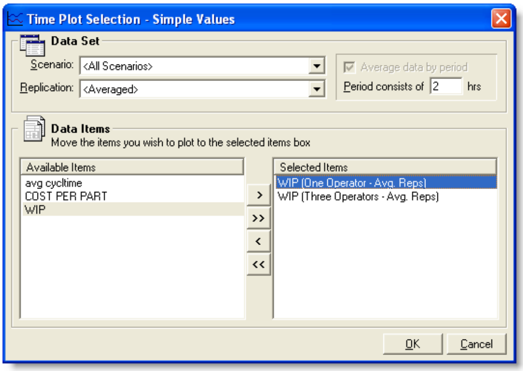
The One Operator scenario additionally displays the minimum and maximum contents of the queue for each time period for all replications, as well as the contents 99.99% confidence interval, which means that 99.99% of all queue contents will be with the two confidence interval lines by period.

This chart shows the impact that an additional operator has on the contents of a queue.

Simple Values Time plot

The simple values time plot shows the change in variable values over time.

Selection Window



Time Plot Selection - Simple Values

Data Set

Scenario: <All Scenarios>

Replication: <Averaged>

☑ Average data by period

Period consists of 2 hrs

Data Items

Move the items you wish to plot to the selected items box

Available Items: avg cycltime, COST PER PART, WIP

Selected Items: WIP (One Operator - Avg. Reps), WIP (Three Operators - Avg. Reps)

Choose the scenarios and replications you would like displayed for the data items selected. Double click on an item to move it from the Available Items area to the Selected Items area or back again. You may also use the arrow buttons to move items between the two areas.

Choose the number of hours in each period that will be displayed along the X-Axis.

Chart Properties

Choosing the properties option for simple values time plot will open a window with two tabs: Data and Display

Chart Properties: Data

The chart properties:data for the simple values time plot are the same as for the time-weighted values histogram. See "Chart Properties" on page 394.

Chart Properties: Display

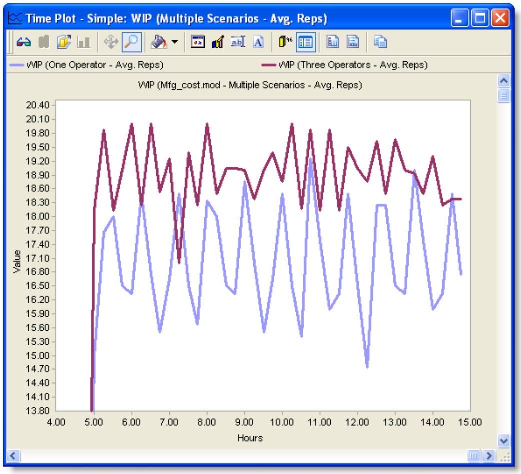
The Display tab for the time-weighted time plot properties window is functionally equivalent as the Category Chart Properties: Display window, with the exception of the “Line Properties” option.

- **Line Properties:** Choose the pattern and thickness for the lines on the chart.

See “Category Chart Properties” on page 392.

Chart Example

The example chart below compares the value of the variable “WIP” over time for two scenarios.

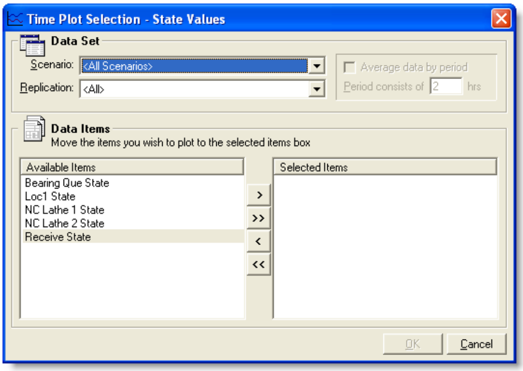


The Y-Axis shows the value of the WIP (work in progress) variable over the time in the X-Axis. The example shows how adding two additional resources to the demo model increases WIP and reduces the fluctuation in WIP.

State Values Time plot

The state values time plot shows location state statistics over time.

Selection Window


Time Plot Selection - State Values

Data Set

Scenario: <All Scenarios>

Replication: <All>

□ Average data by period

Period consists of 2 hrs

Data Items

Move the items you wish to plot to the selected items box

Available Items: Bearing Que State, Loc1 State, NC Lathe 1 State, NC Lathe 2 State, Receive State

Selected Items:

Choose the scenarios and replications you would like displayed for the data items selected. Double click on an item to move it from the Available Items area to the Selected Items area or back again. You may also use the arrow buttons to move items between the two areas.

Chart Properties

Choosing the properties option for state values time plot will open a window with two tabs: Data and Display

Chart Properties: Data

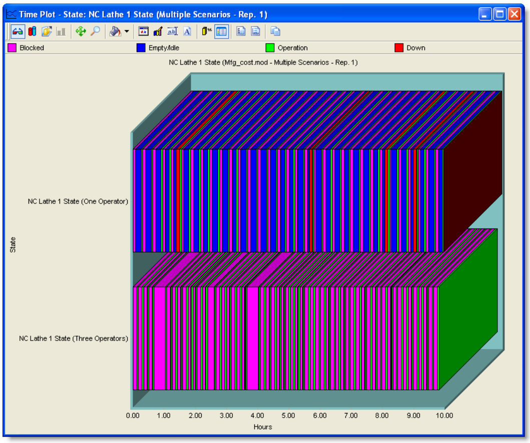
The only option available from this properties window is to change the time units shown in the X-Axis.

Chart Properties: Display

The Display tab for the state values time plot properties window is functionally equivalent as the Category Chart Properties: Display window. See “Category Chart Properties” on page 392.

Chart Example

The example chart below compares the location states of a location for two scenarios.

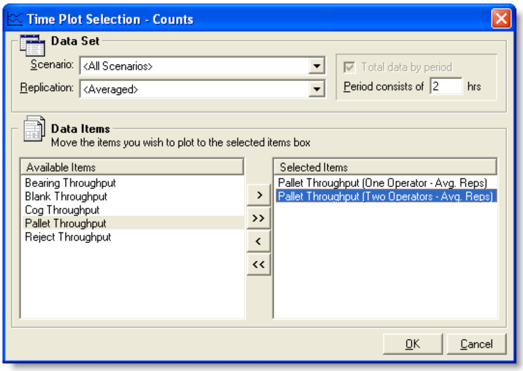


The Y-Axis shows the state of the location for the time in the X-Axis. This example shows how adding two resource units can impact a location's state over time.

Counts Time Plot

The counts time plot shows the number of occurrences of some particular event in the simulation (e.g. the throughput of an entity).

Selection Window



Scenario: <All Scenarios>

Replication: <Averaged>

☑ Total data by period

Period consists of 2 hrs

Move the items you wish to plot to the selected items box

Available Items: Bearing Throughput, Blank Throughput, Cog Throughput, Pallet Throughput, Reject Throughput

Selected Items: Pallet Throughput (One Operator - Avg. Reps), Pallet Throughput (Two Operators - Avg. Reps)

Choose the scenarios and replications you would like displayed for the data item selected. Double click on an item to move it from the Available Items area to the Selected Items area or back again. You may also use the arrow buttons to move items between the two areas.

Choose the number of hours in each period that will be displayed along the X-Axis.

Chart Properties

Choosing the properties option for the counts time plot will open a window with two tabs: Data and Display

Chart Properties: Data

The chart properties:data for the counts time plot are the same as for the time-weighted values histogram. See "Chart Properties" on page 394.

Chart Properties: Display

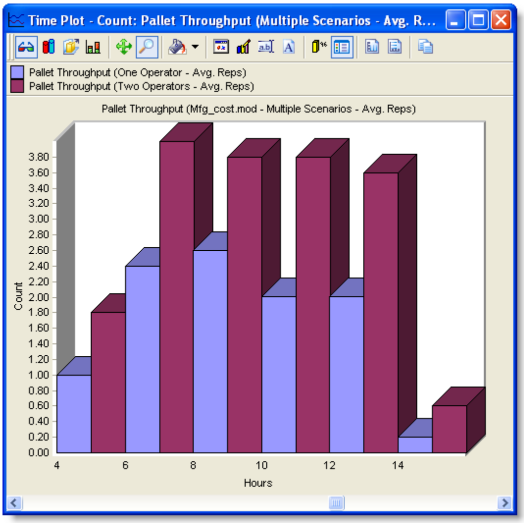
The Display tab for the counts time plot properties window is functionally equivalent as the Category Chart Properties: Display window, with the exception of the “Line Properties” option.

- **Line Properties:** Choose the pattern and thickness for the lines on the chart.

See “Category Chart Properties” on page 392.

Chart Example

The example chart below compares the throughput of a pallet entity for two scenarios.



The X-Axis show time periods. The Y-Axis shows the throughput count for the pallet entity for the time periods in the X-Axis.

This chart shows the improvement in pallet throughput when an extra resource is added to the model.

Chapter 11: Language Elements and Expressions

Language Elements

Language elements are the smallest units of the language used to define a model's objects and logic. Language elements include:

- Names
- Keywords
- Numbers
- Character Strings
- Operators

Names

A name or identifier is any combination (up to eighty characters long) of letters, numbers, and underscores ("_"), used to identify model elements such as locations, entities, variables, and functions. Although any valid name can refer to any object, it is best to use names which describe the object they identify (e.g., using "ClientA" to describe an entity that represents a client). Names, like all words in ProModel, are case *insensitive*, meaning that ProModel sees "PARKING_A," "Parking_A," and "PaRkInG_a" as identical.

Names must use the following conventions:

- Names may contain the letters A through Z (upper or lower case), digits (0-9) and the underscore "_". Names may not contain spaces. When opening an older model, ProModel flags improper use of these restricted characters.
- Do not use a digit as the *first* character of a name. After the first character, any character may be a digit. For example, "2Var" is an invalid name, but "V2", is a valid name.
- Names may not be keywords, but names may contain keywords. For example, "For" is an invalid name, but "*For*klift" is a valid name. Similarly, "*Queue*2" and "*Order*Qty" are both valid.
- No matter what they identify, every name in a model must be unique. For example, you cannot name both a location and an entity "Server." You may, however, name one "Server1" or "Server_Location" and the other "Server2" or "Server_Resource."

Keywords

Keywords are words that ProModel reserves for special use as commands or function calls. Keywords may not be used as names, although names may contain keywords. Keywords, like all words in ProModel, are case insensitive.

Please note

**Reserved for future use.*

For information on tank keywords, see "Tanks" on page 188.

accum
activate
all
alt
and
as
backup
begin
bi
board*
break
breakblk
by
cancel*
calday
calhour
calmin
case*
char
claim*
close
combine
condition
cont
convey
create
day
debug
dec
default
dep
dispatch*
display
do
dosload
drop*
dtleft
else
empty
end
ent
entity
er
exit
fifo
first
for
forlocation
format
forresource
free
full
geo
get
getcost
getresrate
goto
graphic
group
hide*
hr
if
iff
ig
in
inc
incentcost
incloccost
increscost
inf
infinite
int
join
jointly
keep
lifo
load
loc
location
log
lu
maparr
match
min
mod
most
move
move for
move on
move with
ms*
off*
or
order
ownedresource
p5
p6
pause
percentage*
pick*
preemptedres*
preemptor
priority
prompt
queue*
random
read
real
recently*
rename
report
res
reserve*
reset
reset stats
resource
resqty
return
route
sec
send
setrate
show*
skip
snapshot*
sound
split as
stop
take
then
threadnum
timeleft
to
trace
turn
ungroup
unload
until
up*
use
variable
view
wait
warmup
while
wk
write
writeline
xsub
xwrite

Numbers

ProModel uses two types of numbers: real numbers and integers. ProModel also uses a special category of integers, called "name-index numbers." This section discusses real numbers, integers, and name-index numbers. It then discusses converting between the different types.

Integers

An integer number is a whole number ranging from -2,147,483,648 to 2,147,483,647. Integer values may not include commas. Therefore, the number 5,380 should be entered as 5380. Name-index numbers (described in this section) work just like integer numbers.

Examples of integers

-2234798

0

32

Real Numbers

A real number is any number ranging from 1.7 X 10 -308 to 1.7 X 10 +308, including decimals. Real values may not include commas. As such, the number 5,380.5 should be entered as 5380.5.

Examples of real numbers

-2.875638

844.2

65.0

Name-Index Numbers

When a simulation begins, locations, resources, and entities are all assigned numbers according to their position in their respective edit tables. The number assigned to the element is called its *name-index number*. For example, the third entity in the Entity edit table will have the number three for its name index number. The name of an element may be used in an expression to reference its name-index number.

For example, if EntityA were the third entity in the Entity edit table, the statement, "Attr1 = EntityA," would assign the number three to Attr1, because attributes take numbers. Additionally, you may also test for an index number by referencing the element name with a statement like, "IF Var5 = Location3 THEN." In fact, when using a name-index number to identify a location, resource, or entity (as in the previous examples), it is usually best to use the name of the element, because inserting or deleting an element from the edit table, may change the name-index number assigned to other elements.

If you need to refer to a location, resource, or entity by name but only know its name-index number, use the name functions: LOC(), RES(), and ENT(). These functions allow name-index numbers and variables or attributes containing name-index numbers to be converted back to their actual names for use in statements or expressions requiring the element name.

Consider the statement, "Attr1 = PASSENGER." Although this statement only stores the name index number of the element in the attribute, the name-index number can then be used in conjunction with LOC(), ENT(), and RES() to get the actual name of the location. For example, if PASSENGER is the fifth entity in the Entity edit table, the statement LOAD 1 ENT(Attr1) works the same as LOAD 1 PASSENGER. Additionally, a name-index number can be used with the name functions to output a name to the screen or a file.

Converting Between Numeric Types

When an expression expects one type of value but receives another, ProModel automatically takes care of converting between the two, so most often the difference will not matter. However, when ProModel expects an integer but receives a real value, it truncates the real value at the decimal point. For example, say the variable Integer1 is assigned the value 3.9, as in the following statement: Integer1=3.9. The variable Integer1 would hold the value 3, not 4. To round 3.9 when assigning to an integer variable would require the statement, Integer1=Round(3.9).

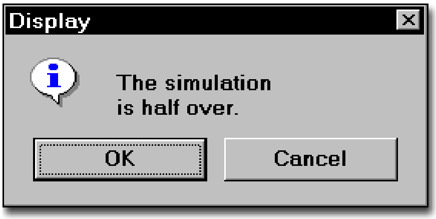
Converting from name-index numbers to the name of an element requires the ENT(), LOC(), and RES() functions as ProModel does not automatically convert numbers to names. For information on how to output the name of an element based on its name-index number, see "String Expressions" on page 412.

Character Strings

A string is any collection of characters enclosed in quotes, such as "Now Boarding." Unlike some programming languages, ProModel uses character strings exclusively for output either to a file or to the screen. Strings may contain any of the 256 ASCII characters. (ASCII characters not on the keyboard can be included in string expressions with the CHAR() function.) Strings may include keywords and names, but those keywords and names will not be executed in or used by the logic.

Additionally, the symbol "\n" can be used to divide a string into multiple lines when it is output, as in the example below.

The statement, DISPLAY "The simulation \nis half over." displays this dialog box:



The simulation
is half over.

Examples of character strings

"EntA exited the system."

"Location 3 is operating at capacity."

"ProModel"

Although ProModel does not provide string variables, it does allow the names of model elements to be stored in variables and attributes as name-index numbers (see "Name-Index Numbers" on page 406). The name referenced by a name-index number may be used in statements and output to the screen or a file with the ENT(), LOC(), and RES() functions. See "String Expressions" on page 412 for more information.

Operators

Operators are symbols used to perform operations on elements in an expression. ProModel operators include Boolean operators, mathematical operators, and a string operator. Unlike reserved words, you may not use operators as any part of a name. The following sections explain the use of these operators for different expression types.

Mathematical Operators

Operator	Meaning	Example
+	Addition	A=B+C
-	Subtraction	A=B-C
*	Multiplication	A=B*C
/	Division	A=B/C
Mod or @	Modulus	A=B Mod C (assigns the remainder of B divided by C to A)
**	Exponentiation	A=B**C (assigns the value of B raised to the C power)

Relational Operators

Operator	Meaning
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

Comparison Operators

Operator	Meaning
AND	TRUE if both expressions are TRUE
OR	TRUE if one expression is TRUE
NOT	Makes TRUE expressions FALSE and FALSE expressions TRUE

Operator Precedence

As in conventional mathematics, ProModel evaluates expressions with more than one operator according to certain rules of precedence. Expressions with more than one operator evaluate in the following order:

1. Terms inside parentheses: ( )
2. Exponentiation: **
3. Multiplication: *; Division: /; and Modulus: @
4. Addition: +; Subtraction: -
5. Equalities and Inequalities: =, <>, >, >=, <, <=
6. NOT
7. AND
8. OR
9. Concatenation: $; (For string expressions only.)

For more information and additional examples of operator precedence, see “Operator Precedence” on page 413.

Expressions

Expressions consist of a value or combination of values (even of different types), variables, attributes, functions, and operators that result in a value. Many consider an expression to be a string expression if it evaluates to a string, a numeric expression if it evaluates to a number, and so on. (For information on how to handle different value types in the same expression, see "Converting Between Numeric Types" on page 407.)

The following are the different expression types used in ProModel:

- Numeric Expressions
- Boolean Expressions
- Time Expressions
- String Expressions

Numeric Expressions

A numeric expression is a combination of numeric elements (such as numbers, variables, and functions) and operators that evaluates to a numeric value.

An expression can contain any of the following in any combination:

Arrays	Mathematical operators
Attributes	Name-Index numbers
Boolean operators	Numbers
Distributions	Subroutines
Macros	System functions
Math function table functions	Variables

The following mathematical operators are available:

Operator	Meaning	Example
+	Addition	A=B+C
-	Subtraction	A=B-C
*	Multiplication	A=B*C
/	Division	A=B/C
Mod or @	Modulus	A=B Mod C (assigns the remainder of B divided by C to A)
**	Exponentiation	A=B**C (assigns the value of B raised to the C power)

You can perform additional mathematical operations using math functions. For more information on operator precedence rules and expression nesting, see "Operator Precedence" on page 413.

Please note

In each of the examples above, if A is an integer number and B and C are both real numbers, the result will be truncated unless ROUND() is used (e.g., A = ROUND(B+C)).

Boolean Expressions

Boolean expressions are relational comparisons between numeric expressions resulting in a value of either True or False. Boolean expressions are used most often in IF...THEN and other control statements. The following relational operators are used in Boolean expressions:

Operator	Meaning
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

Relational operator examples

```
A = B
A > B
A < B
A >= B
A <= B
A <> B
```

Spacing between the terms and operators within a numeric expression is optional. However, for operators with two characters (for example, >=, <=, <>), no spaces are allowed between characters in the operator.

Additionally, the following operators compare two Boolean expressions.

Operator	Meaning
AND	TRUE if both expressions are TRUE
OR	TRUE if one expression is TRUE
NOT	Makes TRUE expressions FALSE and FALSE expressions TRUE

For information on operator precedence and nesting rules, see "Operator Precedence" on page 413.

Time Expressions

A time expression is a numeric expression followed optionally by a time unit (WK, DAY, HR, MIN, SEC) which defines a duration. If no time unit is specified, the default time unit specified in the General Information dialog box is assumed. ProModel knows whether an expression is a time expression based on the context. Time expressions are not valid within a function call. For example, E(10 sec) is invalid, while E(10) sec is valid. If a negative value is used for a time expression, it is automatically converted to zero.

Syntax for time expressions

<numeric expression>{*<time units>*}

Examples

2 hr
E(10) min
Var1+40.25 min.01

Note that time units cannot be mixed (e.g., 3 min 21 sec), unless you use colon notation.

Colon Notation for Time Values

Colon notation allows for more complex time expressions. You may use colon notation in the following manner to express time values.

Colon notation

```
<hours expression>:<minutes expression>:<seconds expression>
```

Colon notated time expressions evaluate from right to left as illustrated in the following examples:

Example	Meaning
:05	5 seconds
5:00: or 5::	5 hours
5:00 or 5:	5 minutes
2:40:	2 hours and 40 minutes
15:02:05	15 hours, 2 minutes and 5 seconds
3:25	3 minutes and 25 seconds
Attr1::Var2	Attr1 hours and Var2 seconds

String Expressions

Any text to be written to a file or displayed on the screen is created with a string expression. String expressions are created from a combination of the following:

- Character strings
- Concatenation operators "$"
- Numeric Expressions
- CHAR()
- ENT()
- LOC()
- RES()
- FORMAT()

A string expression must begin with a concatenation operator ($) or with a string constant in quotes (i.e., "..."). The concatenation operator combines components together, and must be between all components of a string expression. For example, if Attr1 is 7.83, then the expression "STRING1"$ ATTR1 $ "STRING2" evaluates to "String17.83String2."

To output the name of a location, entity, or resource, use the appropriate function inside a string expression: LOC() for locations, ENT() for entities, and RES() for resources. For example, the following logic checks the first ten locations in a model and displays the name of any location that is completely full.

Display name of full location

```
Var1 = 1
While Var1 <= 10 Do
    Begin
        If FreeCap(Loc(Var1)) = 0 Then
        DISPLAY Loc(Var1) $ "is full."
        INC Var1
    End
```

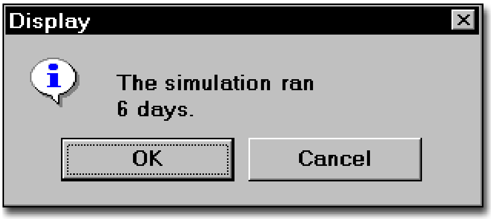
String expressions are valid only as part of the following statements and functions: DISPLAY, PAUSE, PROMPT, STOP, WRITE, WRITELINE, TRACE, XSUB(), and XWRITE.

String expressions

If a model had been running for six days, the statement

DISPLAY "The simulation ran \n" $ CLOCK(DAY) $ "days."

would display the following dialog box.



Display

The simulation ran
6 days.

Operator Precedence

As in conventional mathematics, ProModel evaluates expressions with more than one operator according to certain rules of precedence. Expressions with more than one operator are evaluated in the following order:

1. Terms inside parentheses: ( )
2. Exponentiation: **
3. Multiplication: *; Division: /; and Modulus: @
4. Addition: +; Subtraction: -
5. Equalities and Inequalities: =, <>, >, >=, <, <=
6. NOT
7. AND
8. OR
9. Concatenation: $; (For string expressions only.)

When evaluating more than one operator with the same precedence, ProModel works from left to right.

Arithmetic and boolean expressions may include nested expressions (expressions in parentheses) to indicate precedence in complex expressions such as the following examples:

Nested expressions

A+B*(C+D)

((A>B) OR (B>C)) AND (C>D)

The expression (C+D) is a nested expression which is evaluated first. Multiple nesting is permitted, with the innermost nesting being evaluated first.

Example 1

15 - MATRIX(4)**2 + CLOCK(MIN) / 60

In this expression, Matrix(4)**2 is evaluated first, with the result being subtracted from 15. This value is then added to the value obtained from Clock(min)/60.

Example 2

(A>B) OR (A>C) AND (A=D)

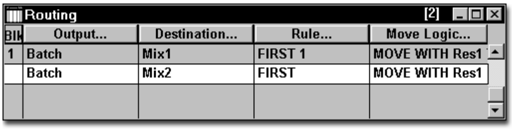
would evaluate

(A>B) OR ((A>C) AND (A=D))

Chapter 12: Routing Rules

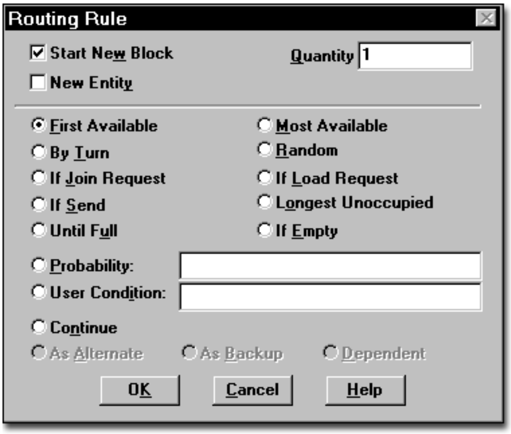
Routing Rules

Routing rules determine the next location for the processing entity. They are specified in the "Rule" field of the Routing edit table. They may be typed manually or selected from the Routing Rule Dialog box shown below by clicking on the Rule button. For more information about routing, see "Routing Edit Table" on page 155.



Routing [2]

Blk	Output...	Destination...	Rule...	Move Logic...
1	Batch	Mix1	FIRST 1	MOVE WITH Res1
	Batch	Mix2	FIRST	MOVE WITH Res1



Routing Rule

☑ Start New Block   Quantity 1
□ New Entity

● First Available   ○ Most Available
○ By Turn   ○ Random
○ If Join Request   ○ If Load Request
○ If Send   ○ Longest Unoccupied
○ Until Full   ○ If Empty
○ Probability:
○ User Condition:
○ Continue
○ As Alternate   ○ As Backup   ○ Dependent

Alternate

Syntax samples

ALT

Description

Causes a location to be selected as an alternate destination if it has available capacity and the condition for a preceding routing rule is not met. If the ALTERNATE location is unavailable, the entity waits until it becomes available or until the preceding routing rule is satisfied.

ALTERNATE routings are common to all primary routings, except the Probability and User Condition rules, and should therefore be listed after the last primary routing but before any backup routings.

For example, a high-speed machine might be preferred, but alternatively, a slower machine can be used when the faster machine is unavailable.

Valid In

The rule field of the Routing edit table only. An ALTERNATE routing may be specified after any other type of routing (including other ALTERNATE routings) except for BACKUP, CONTINUE and DEPENDENT routings.

Example

After Shaft completes a 3 minute operation at Drill, it routes to either Lathe1 or Lathe2 if a SEND statement somewhere in the model directs an EntA to be sent to one of those locations. If no SEND request is waiting, an alternate routing will route the entity to Lathe3. If no capacity is available at Lathe3, the entity will wait until capacity becomes available at Lathe3 or a SEND statement that matches the entity and the location in one of the SEND routings executes somewhere in the model.

Process Table

	Location	Operation (min)
Shaft	Drill	WAIT 3

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Shaft	Lathe1	Send 1	
	Shaft	Lathe2	Send	
	Shaft	Lathe3	ALT	

See Also

See the other routing rules for exceptions and special cases. Also see “Processing” on page 149.

Backup

Syntax samples

BACKUP

Description

Locations specified by this rule are selected only if all of the destinations (primary or alternate) listed in the preceding routings of the same block are unavailable due to any downtime, including a shift. One use of BACKUP is to handle a machine that is not functioning, and entities need another destination to continue.

Valid In

Inside the rule field of the Routing edit table, at the end of a routing block, after one or more primary routings or even an ALTERNATE rule. Conditions for using Backup routings with either USER CONDITION and PROBABILITY routing rules are given in the notes for these rules.

Example

After an entity, Chip, completes a 3 minute operation at CNC, it is routed to Stamp1 when Stamp1 becomes available. Only if Stamp1 and the alternate location (Stamp2) are both down will it be routed to Stamp3. If Stamp3 is also unavailable, Chip will wait for Stamp3 only as long as both Stamp1 and Stamp2 remain down.

Process Table

	Location	Operation (min)
Chip	CNC	WAIT 3

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Chip	Stamp1	FIRST 1	
	Chip	Stamp2	ALT	
	Chip	Stamp3	BACKUP	

See Also

ALTERNATE routing rule. Also see “Processing” on page 149.

Continue

Syntax samples

CONT 1

Description

Leaves an entity at the current location for further processing. ProModel searches the process list forward and then from the beginning until a process is found for the output entity at the current location. If the entity name remains the same and no additional processes for that entity type are defined at the location, the original process will repeat continuously unless a condition bypasses the routing block with the CONTINUE rule.

The CONTINUE routing rule also allows dynamic priority changes. This means a low priority entity that enters a location can be dynamically upgraded to a higher priority without leaving the location. Move Logic for the routing block with the CONTINUE rule will not be executed.

Neither a time value nor a movement rescue is allowed in the Move Logic column for a route block containing a CONTINUE rule. This rule is one way to route an entity from a single capacity location to the same location without causing a deadlock. The CONTINUE rule can also simulate an unlimited supply of a raw material at a location.

Statistics for a location using the CONTINUE rule are collected as one entry for the entity.

Valid In

The rule field of the Routing edit table only. A CONTINUE rule must be the only routing in a routing block and must always take a quantity of one.

Example

This example simulates an unlimited supply of ore at a location. After the Extract location extracts the Iron from the Ore, two routing blocks execute. The first routing block routes the Iron just extracted on to a molding location. The second routing block routes another unit of ore back to the location, where the process begins again.

Process Table

	Location	Operation (min)
Ore	Extract	WAIT 100
Iron	Mold	...

Routing Table

Blk	Output	Destination	Rule	Move Logic
1*	Iron	Mold	FIRST 1	MOVE FOR 10
2	Ore	Extract	CONT 1	
...	...	...	...	...

See the next example for an illustration of dynamic priority changing.

Example

Dynamic Priority Changing Suppose a gear enters a location. The gear has a low priority. If another type of gear enters the system with a higher priority, the second gear may preempt the first gear. Another worker in the plant needs the first gear immediately or else he will miss his deadline. Now the first gear's priority is higher than the second gear. Using the CONTINUE routing rule allows you to change the priority of the first gear.

An entity that enters a location with a priority 99 can increase its priority to 999 without ever leaving that location. In addition, this functionality will also allow you to specifically control which entities in a multi-capacity location will be preempted by another incoming high priority entity.

Process Table

	Location	Operation (min)
Gear	Loc1	WAIT 2

Gear	Loc2	WAIT 3
		IF CLOCK() > 1 THEN
		  BEGIN
		    var = 999
		    ROUTE 2
		  END
		ELSE ROUTE 1

Gear	Loc2	

Routing Table

	Output	Destination	Rule	Move Logic
1	Gear	Loc2, 99	FIRST 1	MOVE FOR 1
1	Gear	Loc3	FIRST 1	MOVE FOR 1
2	Gear	Loc2, var	CONT 1	
1	Gear	Loc3	FIRST 1	MOVE FOR 1

See Also

“Processing” on page 149.

Dependent

Syntax samples

DEP

Description

Selects a location if, and only if, the routing immediately preceding it is selected. An ALTERNATE routing which follows a dependent routing is an alternate to the last major routing preceding the dependent condition and not to the dependent routing itself.

A dependent routing should be used when one process results in two different types of entities that should go to different locations. For example, a dependent routing could simulate the separation of a customer and his or her order placed at the drive-through window at a restaurant. While the customer continues on to the cashier window, the order is sent on to the counter for fulfillment.

Valid In

The rule field of the Routing edit table only. A DEPENDENT routing may be specified after any other routing rule (including other DEPENDENT routings) except for a CONTINUE routing.

Example

After an entity named "Lamp" completes a 3-minute preparation at Prep, the paint shop will paint it. If the paint shop has capacity, the Base routes to the paint shop and the Shade routes to the Waiting area based on the dependent rule. But if the paint shop does not have capacity, the lamp stays together and routes to Storage on an alternate routing.

Process Table

	Location	Operation (min)
Lamp	Prep	WAIT 3

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Base	Painting	FIRST 1	
	Shade	Waiting	DEP	
	Lamp	Storage	ALT	

See Also

"Processing" on page 149.

Empty

Syntax samples

EMPTY {<*expression*>}

Description

Selects a location only if it is completely empty. This routing rule is similar to the UNTIL FULL rule except that a location must be completely empty before it is initially selected. Once an empty location is selected, it continues to be selected until it is full. If no location is empty, the output waits until one becomes empty.

The EMPTY condition is designed for the situation where two or more multi-capacity locations are being filled from the same source and the modeler desires each location to completely fill and completely empty in an alternating fashion.

Valid In

The rule field of the Routing edit table only.

Components

<expression>

Total entities output from the process. This expression is valid only for the first routing of a routing block. For more information on this expression, see "Processing" on page 149.

Example

After EntA completes a 25 second operation at Loc1, it is routed to one of three multi-capacity locations (Loc2, Loc3 or Loc4) as soon as one becomes empty. Subsequent routings continue to the same location until it is full, and then to the next location that is empty. Once all locations are filled, no more routings occur until one of the locations becomes empty.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT 25 sec

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	EMPTY 1	
	EntA	Loc3	EMPTY	
	EntA	Loc4	EMPTY	

See Also

"Processing" on page 149.

First Available

Syntax samples

FIRST {<*expression*>}

Description

Selects the first location available among one or more locations listed in a routing block. Specifying multiple First Available routings in a routing block has the same effect as specifying a First Available routing followed by one or more ALTERNATE routings. An example of using the First Available rule in a routing block is given below.

The First Available rule is the default rule when defining the initial routing in the Process Editor. In this case, the First Available routing can be interpreted to be the "primary" routing.

Valid In

The rule field of the Routing edit table only.

Components

<expression>

The total entities output from the process. This expression is valid only for the first routing of a routing block. For more information on this expression, see the "Processing" on page 149.

Example

After EntA completes a 4 minute operation at Loc1, it is routed to Loc2, Loc3 or Loc4 depending on which location is the first to have available capacity.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT 4

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	FIRST 1	
	EntA	Loc3	FIRST	
	EntA	Loc4	FIRST	

See Also

"Processing" on page 149.

Join

Syntax samples

JOIN {<*expression*>}

Description

Selects a location whenever a JOIN request is issued at that location. Since a joining entity does not require capacity, there is no need to check for available capacity at the destination. Multiple JOIN requests are filled according to the oldest waiting request with the highest priority. Entities routed with the JOIN routing rule are not actually sent to their destination until a JOIN statement is encountered at the destination location.

Valid In

The rule field of the Routing edit table only.

Components

<expression>

The total entities output from the process. This expression is valid only for the first routing of a routing block. For more information on this expression, see "Processing" on page 149.

Example

After EntA completes a 5.2 minute operation at Loc1, it routes to Loc2, Loc3 or Loc4 to join to some other entity that issued a JOIN request for an EntA.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT 5.2

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	JOIN 1	
	EntA	Loc3	JOIN	
	EntA	Loc4	JOIN	

See Also

JOIN statement. Also see "Processing" on page 149.

Load

Syntax samples

LOAD {<*expression*>}

Description

Selects a location whenever a LOAD request is issued at the destination location. Since a loading entity does not fill capacity, there is no need to check for available capacity at the destination. Multiple LOAD requests are filled according to the oldest waiting request with the highest priority.

Valid In

The rule field of the Routing edit table only.

Components

<expression>

The total entities output from the process. This expression is valid only for the first routing of a routing block. For more information on this expression, see “Processing” on page 149.

Example

After EntA completes a 3 minute operation at Loc1, it is routed to Loc2, Loc3 or Loc4 to be loaded onto some other entity that has issued a LOAD request.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT 3

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	LOAD 1	
	EntA	Loc3	LOAD	
	EntA	Loc4	LOAD	

See Also

LOAD statement. Also see “Processing” on page 149.

Longest Unoccupied

Syntax samples

LU {<*expression*>}

Description

Selects one of the locations listed in a block of routings based on which has been unoccupied the longest. If several multi-capacity locations all have one or more current entities, the location with the most available capacity will be selected. If no capacity is available at any location, the first one that becomes available is selected. This rule is useful in situations where residual effects must diminish before further usage of a location (e.g., an oven cooling to an ambient temperature, the vapor clearing out of a paint booth, etc.).

Please note that this routing rule is not valid for single-capacity locations

Valid In

The rule field of the Routing edit table only.

Components

<expression>

The total entities output from the process. This expression is valid only for the first routing of a routing block. For more information on this expression, see "Processing" on page 149.

Example

After EntA completes a 2.5 minute operation at Loc1, it routes to Loc2, Loc3 or Loc4 depending on the location unoccupied the longest.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT 2.5

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Plane	Gate1	LU 1	MOVE FOR 2
	Plane	Gate2	LU	MOVE FOR 2
	Plane	Gate3	LU	MOVE FOR 2
	Plane	Gate4	LU	MOVE FOR 2

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	LU 1	
	EntA	Loc3	LU	
	EntA	Loc4	LU	

See Also

"Processing" on page 149.

Most Available

Syntax samples

MOST {<*expression*>}

Description

Selects one of the locations listed in a block of routings based on which has the most available capacity. If no capacity is available at any of the locations listed, the first one that becomes available is selected. Use the MOST routing rule to equalize queues in front of workers. This rule is useful in cases where inventory levels need to be balanced among several downstream queues.

Valid In

The rule field of the Routing edit table only.

Components

<expression>

The total entities output from the process. This expression is valid only for the first routing of a routing block. For more information on this expression, see "Processing" on page 149.

Example

After EntA completes a 4 minute operation at Loc1, it is routed to Loc2, Loc3 or Loc4 depending on which location has the most available capacity at the current time.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT 4

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	MOST 1	
	EntA	Loc3	MOST	
	EntA	Loc4	MOST	

See Also

"Processing" on page 149.

Probability

Syntax samples

<probability> {*<expression>*}

Description

Randomly selects a location listed in a block of routings based on a probability. Several probability routings should be used together and the sum of all probabilities must equal one. The entity will wait to be routed until the selected location has available capacity. Unlike most primary routings, an ALTERNATE and BACKUP routing may be specified after each PROBABILITY routing in a single routing block. If the selected location has no ALTERNATE routing and has no available capacity, the entity will wait for the location until it has available capacity.

Valid In

The rule field of the Routing edit table only.

Components

<probability>

The chance of the routing being taken, expressed as a decimal less than one. All probabilities in a single routing block must add up to one. This number must be a value and may not be an expression.

<expression>

The total entities output from the process. This expression is valid only for the first routing of a routing block. For more information on this expression, see "Processing" on page 149.

Example

EntA completes an eight minute operation at Loc1, and routes to Loc2 80% of the time, Loc3 15% of the time, and to Loc4 5% of the time. If EntA selects Loc2 but the location has no available capacity, EntA will select an alternate location, Loc2A. If both Loc2 and Loc2A are down, EntA selects a backup location (Loc2B). Loc3 has an alternate location (Loc3A) in case it is unavailable, but has no backup location. Loc4 has neither an alternate location nor a backup location.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT8

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	.800 1	
	EntA	Loc2A	ALT	
	EntA	Loc2B	BACKUP	
	EntA	Loc3	.150	
	EntA	Loc3A	ALT	
	Claim	Loc4	.050	

See Also

"Processing" on page 149.

Random

Syntax samples

RANDOM {<*expression*>}

Description

Randomly selects one of several available locations listed in a block of routings such that each location having available capacity is equally likely to be selected. If none of the locations listed has available capacity, the first location that becomes available will be selected.

Valid In

The rule field of the Routing edit table only.

Components

<expression>

The total entities output from the process. This expression is valid only for the first routing of a routing block. For more information on this expression, see "Processing" on page 149.

Example

After EntA completes a 3 minute operation at Loc1, it is randomly routed to Loc2, Loc3, or Loc4.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT 3

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	RANDOM 1	
	EntA	Loc3	RANDOM	
	EntA	Loc4	RANDOM	

See Also

PROBABILITY routing rule. Also see "Processing" on page 149.

Send

Syntax samples

SEND {<*expression*>}

Description

Causes an entity to remain at the current location until one of the listed destinations issues a SEND. Once a location has been selected, capacity must be available at that location before the routing actually takes place. Multiple alternative SEND destinations may be specified in a block.

Valid In

The rule field of the Routing edit table only.

Components

<expression>

The total entities output from the process. This expression is valid only for the first routing of a routing block. For more information on this expression, see "Processing" on page 149.

Example

Purchase Orders (PO's) are held at a holding location until final approval is received. Once a SEND request has been generated at some other location in the system, the PO is sent to the appropriate vendor.

Process Table

	Location	Operation (min)
PO	Holding	

Routing Table

	Output	Destination	Rule	Move Logic
1	PO	VendorA	SEND 1	MOVE FOR 48 Hr
	PO	VendorB	SEND	MOVE FOR 48 Hr
	PO	VendorC	SEND	MOVE FOR 48 Hr

See Also

"Processing" on page 149.

Turn

Syntax samples

TURN {<*expression*>}

Description

Selects the locations listed in a block of routings in rotation by availability. If none of the locations listed are available, the first one that becomes available is selected. A particular location may be listed more than once in a routing block if it is to have a greater proportion of turns than the others.

Valid In

The rule field of the Routing edit table only.

Components

<expression>

The total entities output from the process. This expression is valid only for the first routing of a routing block. For more information on this expression, see “Processing” on page 149.

Example

In this example, Purchase Requests (PR’s) are entered into an on-line accounting system at location Keypunch. They are then assigned on a rotating basis to Buyer1, Buyer2, or Buyer3.

Process Table

	Location	Operation (min)
PO	Keypunch	WAIT U(3,1) min

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	PR	Buyer1	TURN 1	
	PR	Buyer2	TURN	
	PR	Buyer3	TURN	

See Also

“Processing” on page 149.

Until Full

Syntax samples

FULL {<*expression*>}

Description

This rule continues to direct all output to the first location specified until it fills to capacity and then to the next location until it fills and so on. If all locations are full, the first one that becomes available is selected.

Valid In

The rule field of the Routing edit table only.

Components

<expression>

The total entities output from the process. This expression is valid only for the first routing of a routing block. For more information on this expression, see “Processing” on page 149.

Example

In this example, pallets are scanned at a barcode location before being placed in storage in a warehouse. The policy at the warehouse calls for keeping Aisle1 completely full before storing anything in Aisle2. Similarly, Aisle2 must be kept full before storing anything in Aisle3.

Process Table

	Location	Operation (min)
Pallet	Barcode	USE Scanner FOR U(2.3,4)

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Pallet	Aisle1	FULL 1	MOVE FOR 5
	Pallet	Aisle2	FULL	MOVE FOR 6
	Pallet	Aisle2	FULL	MOVE FOR 7

See Also

“Processing” on page 149.

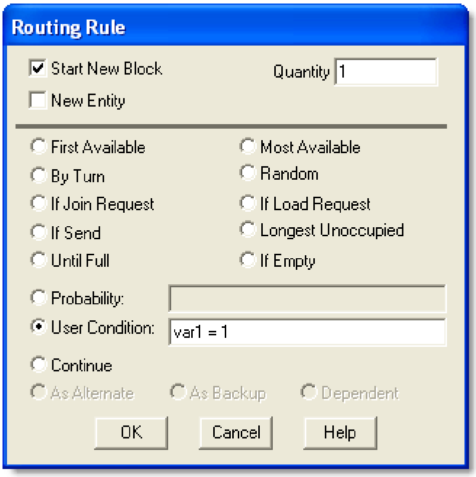
User Condition

Syntax samples

<Boolean expression> {<expression>}

Description

Selects one of several locations listed in a routing block based upon a user-defined condition. Several User-Condition routings usually are used in the same routing block. At least one of the user-defined conditions in a block must be true or an error message will appear and the simulation will terminate. Capacity must be available at the location before the routing actually takes place. Unlike most primary routing rules, an ALTERNATE or BACKUP routing may be specified after each USER CONDITION routing in the same routing block.



Routing Rule

☑ Start New Block Quantity 1

□ New Entity

○ First Available ○ Most Available

○ By Turn ○ Random

○ If Join Request ○ If Load Request

○ If Send ○ Longest Unoccupied

○ Until Full ○ If Empty

○ Probability:

● User Condition: var1 = 1

○ Continue

○ As Alternate ○ As Backup ○ Dependent

Valid In

The rule field of the Routing edit table only.

Components

<Boolean expression>

Any expression which evaluates to TRUE or FALSE.

<expression>

The total entities output from the process. This expression is valid only for the first routing of a routing block. For more information on this expression, see “Processing” on page 149.

Example

Customers bring their cars to a local service station for emissions testing and state inspection. After each car completes a 10-minute inspection operation at location Inspect, they are put in the proper service bay depending on the car’s engine type. Late-model (post-1975) cars are tested in Bays 1 and 2, pre-1975 cars are tested in Bay3, and diesel engine cars are tested in Bay4. If Bay1 is selected but not available, an alternate location (Bay2) is selected. If both Bay1 and Bay2 are down, a backup location (Bay3) is selected. Bay3 also has an alternate location (Bay4) in case it is unavailable, but has no backup location. Bay4 has neither an alternate location nor a backup location.

Process Table

	Location	Operation (min)
CAR	Inspect	WAIT 10

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	CAR	Bay1	IF Type=1 1	
	CAR	Bay2	ALT	
	CAR	Bay3	BACKUP	
	CAR	Bay3	IF Type=2	
	CAR	Bay4	ALT	
	CAR	Bay4	IF Type=3	

See Also

"Processing" on page 149.

Chapter 13: Logic Elements

Functions

Functions return information such as a location's capacity, the amount of time the simulation has been running, or a random number. A function is a keyword followed by parentheses, which may contain data that the function needs to complete its operation. You should be aware that if a function or variable expecting an integer gets a real number, it will ignore everything to the right of the decimal point.

System Functions

The following list describes the function types available in ProModel.

General System Functions

General system functions return information about the simulation, such as the length of time that the simulation has run, and may be referenced in any numeric expression.

Entity-Specific System Functions

Entity-specific system functions return information about the processing entity, such as the number of entities in an entity group. For that reason, these functions may be used only where an entity is processing some type of logic. This includes Arrival, Operation, and Location Exit logic.

Resource-Specific System Functions

Resource-specific system functions may only be used in resource node logic when a resource enters or leaves a node.

Downtime-Specific System Functions

ProModel permits you to use downtime-specific system functions only in downtime logic.

Shift & Break System Functions

ProModel allows you to use these functions only in shift and break logic.

General Functions

Math Functions

Math functions are built-in functions used for performing mathematical operations on numeric expressions.

Type Conversion Functions

When two types of numbers are used in an expression, as when a real value is assigned to an integer variable, they must all be converted to the same type before ProModel can evaluate them. ProModel performs this conversion automatically and, in most instances, this conversion will be satisfactory. The only place where the user may want to use a different conversion is when ProModel converts from a real to an integer.

When converting from a real to an integer, ProModel ignores everything to the right of the decimal point. When converting from an integer to a real, ProModel simply adds a decimal point to the integer. ProModel handles name-index numbers exactly like integers. Most often, these automatic conversion will be sufficient, but occasionally it will be necessary to convert between the types differently. For these situations, use the conversion functions.

Statements

Statements cause ProModel to take some action or perform some operation. Unlike functions, statements neither return a value nor use parentheses, and logic may contain comments. (See "Comments" on page 461.) Statements can use spacing (including a new line) before each word in a statement, and are case insensitive so any letter may be either upper or lower case.

General Action and Control Statements

General statements can be divided into two main categories: action and control statements. They are called general because they can be used in any logic. Action statements cause some action to occur in the model, such as changing an entity's graphic or writing to a file. Control Statements determine the next statement to be executed, such as loops, branches, and statement blocks.

Resource- and Entity-Related Operation Statements

Operation Statements perform specific actions on entities and resources at locations throughout the system whenever they are encountered by an entity in the operation logic. They are valid only in certain areas. See each statement for a list of places where it is valid.

There are two subsets of operation statements: entity-related and resource-related. Entity-related operation statements perform specific actions on entities only. Resource-related operation statements involve resources alone (for example, GET and FREE) or resources and entities together (for example, USE).

Statement Blocks

A statement block is a group of statements that begin with the keyword BEGIN, or the symbol "{", and end with the keyword END, or the symbol "}". Two examples appear below. See BEGIN and END for more extensive examples of statement blocks.

Statement blocks

```
BEGIN                  {
    Statement 1        Statement 1
    Statement 2        Statement 2
    Statement 3        Statement 3
    ...                ...
    Statement n        Statement n
            END        }
```

Distribution Functions

Distribution functions are built-in functions which, in conjunction with streams, return random values according to a statistical distribution. The following table is a summary of available distribution functions. They are valid in any numeric expression.

Distribution	Syntax	Individual Components
Beta	B(a,b,c,d{,<s>})	a=shape value 1, b=shape value 2, c=lower boundary, d=upper boundary
Binomial	BI(a,b{,<s>})	a=batch size, b=probability of “success”
Erlang	ER(a,b{,<s>})	a=mean value, b=parameter
Exponential	E(a{,<s>,<ax>})	a=mean
Gamma	G(a,b{,<s>,<ax>})	a=shape value, b=scale value
Geometric	GEO(a{,<s>})	a=probability of “success”
Inverse Gaussian	IG(a,b{,<s>,<ax>})	a=shape value, b=scale value
Lognormal	L(a,b{,<s>,<ax>})	a=mean, b=standard deviation
Normal	N(a,b{,<s>})	a=mean, b=standard deviation
Pearson5	P5(a,b{,<s>,<ax>})	a=shape value, b=scale value
Pearson6	P6(a,b,c{,<s>,<ax>})	a=shape value 1, b=shape value 2, c=scale value
Poisson	P(a{,<s>})	a=quantity
Triangular	T(a,b,c{,<s>})	a=minimum, b=mode, c=maximum
Uniform	U(a,b{,<s>})	a=mean, b=half range,
User-defined	<name>({<s>})	Name of a user-defined distribution as defined in the User Distribution section
Weibull	W(a,b{,<s>,<ax>})	a=shape value, b=scale value

General Components

<s> The optional stream to use in conjunction with the distribution’s probabilities. If this option is omitted, ProModel will use stream one. For more information on streams, see “Streams” on page 266.

<ax> An optional axis shift. The distributions, E, G, W, L, IG, P5, and P6 all normally have a minimum value of zero. Use an axis shift to alter the distribution’s minimum value and at the same time shift the entire distribution. Any time an axis shift is specified, a stream must be specified also.

Any negative value returned by a distribution that is used for a time expression will be automatically converted to zero.

Priorities

Many statements in this manual use <priority> as a component. A priority determines the order in which competing commands are fulfilled. For example, priority determines which of two competing requests to USE a resource gets fulfilled, or which of two competing requests to SEND the same entities to different locations gets fulfilled. Commands or entities with high enough priority can even bring back up locations that are down, or force a location to stop processing one entity and process another instead. This is called "preemption." All priorities must be an expression that evaluates to a number between 0 and 999.

For a complete explanation of priorities and preemption, see "Location Priorities and Preemption" on page 111, "Resource Shift Downtime Priorities" on page 141, "Preemptive Entities" on page 121, and "Preemption Process Logic" on page 300.

Chapter 14: Statements and Functions

Accum

Entity-Related Operation Statement

Syntax samples

ACCUM <*expression*>
ACCUM 10
ACCUM Var1

Description

Accumulates, without consolidating, the specified quantity of entities at a location. ACCUM works like a gate that prevents entities from processing until a certain number arrive. Once the specified number of entities have accumulated, they will go through the gate and begin processing individually, independent of each other.

ACCUM can be used to model situations where several entities should be accumulated before they get processed. For example, when a resource processes orders at a work station, it may be more efficient to accumulate several orders before requesting the resource.

If you specify an ACCUM operation in a process for an individual entity, accumulation will occur by individual entity type. However, if you specify ALL as the processing entity, all entity types at that location will participate in the same accumulation.

Valid In

The operation column of process edit tables only. ACCUM must be used at a location with enough capacity to accumulate the specified quantity.

Components

<expression>

The number of entities to accumulate. If this expression results in zero or one, it is ignored. If it results in a number greater than the location's capacity or a negative number, the simulation will stop with an error.

This expression is evaluated every time an entity to be accumulated arrives, so the quantity to be accumulated can vary as the entities to be accumulated arrive. If an entity arrives that changes this expression to a number lower than the number of entities already accumulated, all of the accumulated entities begin to process.

Example

Entities named "Pallet" arrive at a location named "Loading" and accumulate in batches of 20. Entities named "Box" also arrive at Loading and accumulate in batches of 10. Only after the right number of boxes or pallets accumulates does ProModel request the forklift. The accumulation of Boxes and Pallets at Loading is entirely independent because the accumulation statements are in different records. Note that the forklift is used for two minutes per pallet or box, because each pallet and box process individually after accumulating all of them. While the accumulation of entities at Loading is by entity type, the accumulation at the location, "Storage" is independent of entity type since the ALL keyword denotes common processing for all entities at this location.

Process Table

	Location	Operation (min)
Pallet	Loading	ACCUM 20 USE ForkliftFOR 2
Box	Loading	ACCUM 10 USE Forklift FOR 2
ALL	Storage	ACCUM 30

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Pallet	Storage	FIRST 1	MOVE FOR 5
1	Box	Storage	FIRST 1	MOVE FOR 5
1	ALL	Loc3	FIRST 1	MOVE FOR 5

See Also

COMBINE, JOIN, and GROUP.

Activate

General Action Statement

Syntax samples

ACTIVATE *<subroutine>({parameter1>,<parameter2>...})*
ACTIVATE Sub1()

Description

Starts an independent subroutine. Only subroutines of type interactive can be called with ACTIVATE. The calling logic then continues without waiting for the called subroutine to finish. Therefore, independent subroutines can run in parallel with the logic that called them. Independent subroutines are not entity or location dependent and run without regard to what happens inside the logic that called them.

Use ACTIVATE to process logic that has WAIT or WAIT...UNTIL statements when you do not want to use an entity to process the WAIT or WAIT...UNTIL statements. For example, an ACTIVATE in the initialization logic could call a subroutine that adjusts the arrival frequency for a type of entity depending on the time of day.

Independent subroutines called with ACTIVATE cannot use entity-specific or location-specific system functions. If the subroutine has a return value, then that value is ignored. External subroutines cannot be called with ACTIVATE, although they may be called from within an activated subroutine.

Valid In

Any logic.

Components

<subroutine>

The name of the subroutine to run. This name should appear exactly as if the subroutine were being called normally. Any return value for this function is ignored. See “Subroutines” on page 246.

<parameters>

The parameters that the subroutine normally takes.

Example

This example uses ACTIVATE in a model’s initialization logic to start a subroutine named Res_Log(). Res_Log() is a user-defined subroutine that logs every time that all units of a resource named Worker are in use. After it logs the time that all units were busy, it waits ten minutes to check again. Note that the WHILE...DO loop in the subroutine is never exited. This technique allows the subroutine to run during the entire simulation.

Initialization Logic:

```
Activate Res_Log()
```

Res_Log()

```
INT X = 1
WHILE X = 1 DO
BEGIN
IF FREEUNITS(Worker)=0
THEN LOG "All workers busy at ",0
WAIT 10
END
```

See Also

XSUB(). Also see “Subroutines” on page 246.

Animate

General Action Statement

Syntax samples

ANIMATE <*expression*>
ANIMATE 70
ANIMATE Var1

Description

Sets the simulation's animation speed. The higher the value, the faster the animation. ANIMATE is used primarily to speed up or slow down a model for cursory or detailed observation whenever a particular condition is encountered. Another common use is to set the animation speed to one-hundred in the Initialization Logic to rapidly advance the simulation to some point in time.

Valid In

Any logic.

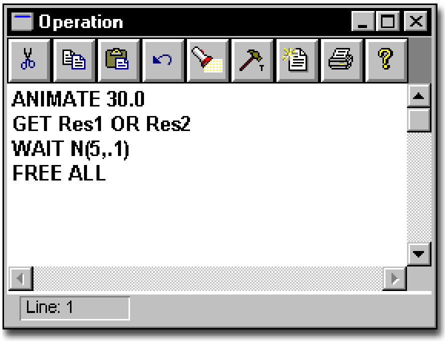
Components

<expression>

The new speed of the animation. This expression can be any number from 0 to 100 and is evaluated every time the ANIMATE statement is encountered. The default speed is 56.

Example

This example shows the use of the ANIMATE statement in the downtime logic for the location Press. Whenever Press goes down for service, the animation speed is slowed to 30 percent of maximum. This allows the modeler to more easily view which resource (RES1 or RES2) is captured to service the location.



```
ANIMATE 30.0
GET Res1 OR Res2
WAIT N(5,.1)
FREE ALL
```

See Also

GRAPHIC and SOUND.

ArrayDims()

General System Function

Syntax samples

ARRAYDIMS(<arrayname>)

ARRAYDIMS(MyArray)

Description

Returns the number of dimensions in an array.

Valid In

Any Logic

Components

<arrayname>

The name of the array for which you wish to know the number of dimensions.

See Also

ArrayDimSize().

ArrayDimSize()

General System Function

Syntax samples

```
ARRAYDIMSIZE(<arrayname>, <dim_num>)
ARRAYDIMSIZE(MyArray, 2)
```

Description

Returns the size of a dimension in an array. You must provide the name of the array and the particular dimension of the array for which you want to know the size.

Valid In

Any Logic

Components

<arrayname>

The name of the array.

<dim_num>

The number of the dimension for which you wish to know the size.

See Also

ArrayDims().

Assignment Statement

Syntax samples

```
<variable, array element, or attribute> =
<numeric expression>
```

Description

Assigns the value of a numeric expression to a designated variable, array element, or attribute.

Valid In

Any Logic

Example

```
Var1 = 300
Attr2 = Clock(hr) - Attr3
```

Please Note

If you assign an expression that evaluates to a real number to a variable, array, or attribute of type integer, the number truncates unless you use the ROUND() function.

Begin

General Control Statement

Syntax samples

```
BEGIN or {


WHILE FREECAP(Loc1) > 5 DO
     BEGIN
          INC Var2, 5
          WAIT 5 sec
     END


IF Var1 > 5 THEN
     {
     INC Var2, 5
     WAIT 5 sec
     }
```

Description

Defines a statement block with a corresponding END statement. BEGIN and END are almost always used in conjunction with other control statements such as IF...THEN and DO...WHILE. Every BEGIN statement must pair with an END statement.

Valid In

Any logic.

Example

Compare the following logic examples:

The example below includes a BEGIN and END statement. In this logic, if the attribute Attr1 equals one, ten entities are ordered and the variable Var1 increments. ProModel executes the statements within the BEGIN and END only if Attr1 equals one.

```
IF Attr1 = 1 THEN
BEGIN
ORDER 10 EntA
INC Var1
END
```

Just as in the logic above, if Attr1 in the following example equals one, ten entities are ordered. However, Var1 increments regardless of the value of Attr1. The IF...THEN applies only to the very next statement, no matter how you format the logic.

```
IF Attr1 = 1 THEN
ORDER 10 EntA
INC Var1
```

See Also

END, IF...THEN...ELSE, DO...WHILE, WHILE...DO, and DO...UNTIL.

Break

General Control Statement

Syntax samples

```
BREAK
```

Description

Exits the innermost WHILE...DO, DO...WHILE, or DO...UNTIL loop. The next statement to be executed will be the one immediately following the END statement associated with the innermost loop. If a BREAK is encountered outside any loop, then ProModel exits the entire logic.

Valid In

Any logic.

Example

Normally, a WHILE...DO loop repeats a statement block until a certain condition becomes false. This example shows the use of a BREAK statement to exit the loop based on a condition not included in the WHILE...DO loop. As long as the variable V1 is less than three and the variable V3 is less than or equal to five, both the three-minute wait and the five-minute wait will be executed for a total of eight minutes. However, if V3 becomes greater than five, the total wait will be six minutes: three minutes in the main part of the loop and another three minutes inside the IF...THEN statement block. The five minute wait will not be executed because the BREAK statement exits the loop entirely.

The line "//Break to here" is a comment that marks the next line of logic executed after the BREAK.

Process Table

	Location	Operation (min)

Claim	Station1	WHILE V1<3 DO
		  BEGIN
		    WAIT 3 min
		    IF V3<=5 THEN
		      BEGIN
		        WAIT 3 min
		        BREAK
		      END
		    WAIT 5 min
		  END
		//Break to here

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Station2	FIRST 1	MOVE FOR 5

See Also

BREAKBLK.

BreakBlk

General Control Statement

Syntax samples

```
BREAKBLK
```

Description

Exits from the innermost statement block. The next statement to be executed will be the one immediately following the END statement of the innermost statement block. If a BREAKBLK is executed outside any statement block, ProModel will exit the logic completely.

Valid In

Any logic.

Example

This example uses BREAKBLK in the logic for a shipping department that handles two types of entities, Letters and Packages. Both packages and letters need to be addressed, but packages may need to be wrapped before they are addressed. When a package arrives, shipping increments the variable No_of_Pkg to keep track of the number of packages it ships. Attr1 is then checked to see if the package has been wrapped. If the package has been wrapped (if Attr1 = 1), the BREAKBLK statement exits the statement block that wraps packages. If the package has not been wrapped, a resource, Wrapper, wraps the package. Finally, all entities that Shipping processes are addressed by the same statement and routed to the location Post_Office. The line
"// Breakblk to here" is a comment which marks the next statement executed after the BREAKBLK.

Process Table

	Location	Operation (min)

ALL	Shipping	IF ENTITY() = Letter THEN
		INC No_of_Let
		IF ENTITY() = Pkg THEN
		BEGIN
		INC No_of_Pkg
		IF Attr1 = 1 THEN
		BREAKBLK
		USE Wrapper FOR 8
		Attr1 = 1
		END
		//Breakblk to here
		USE Addresser FOR 3

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	ALL	Post_Office	FIRST 1	MOVE FOR 5

See Also

BREAK.

CalDay()

General System Function

Syntax samples

```
CALDAY()
```

Description

The CALDAY() function corresponds to the weekday of the calendar date you defined as part of the warm-up period or simulation begin date under simulation options. Since CALDAY() resets with the advent of a new week, every weekday will return the same value (i.e., Wednesday will always return a value of 3).

Valid In

Any logic.

Please note

CALDAY() works only when you select calendar date in the simulation options dialog.

Example

Suppose that shipment arrival types vary from day to day. Since Monday's Shipment_Type_A arrival patterns are unique, you will need to use a distribution specific to those patterns.

```
IF CALDAY()=1 THEN
      BEGIN
          Shipment_Type_A =
          Monday_Distribution_A
          USE Forklift FOR Unload_Time
      END
```

See Also

CALDOM(), CALHOUR(), CALMIN(), CALMONTH(), and CALYEAR().

CalDOM()

General System Function

Syntax samples

CALDOM()

Description

The CALDOM() function corresponds to the calendar day of the month you defined as part of the warm-up period or simulation begin date under simulation options. Values returned by this function will be integers in the range of 1 to 31.

Valid In

Any logic.

Please note

CALDOM() works only when you select calendar date in the simulation options dialog.

See Also

CALDAY(), CALHOUR(), CALMIN(), CALMONTH(), and CALYEAR().

CalHour()

General System Function

Syntax samples

```
CALHOUR()
```

Description

The CALHOUR() function corresponds to the hour of the calendar date you defined as part of the warm-up period or simulation begin date under simulation options. Since this function ties directly to the 24-hour clock displayed on the screen during simulation, CALHOUR() will never return a value higher than 23.

Valid In

Any logic.

Please note

CALHOUR() works only when you select calendar date in the simulation options dialog.

Example

The following example implements a change in personnel used to perform an operation. After the simulation runs for 18 hours, the technician becomes available to perform the activity.

```
IF CALHOUR() > 18 THEN

USE Worker FOR N(10,4)

ELSE USE Technician FOR N(8,2)
```

See Also

CALDAY(), CALDOM(), CALMIN(), CALMONTH(), and CALYEAR().

CalMin()

General System Function

Syntax samples

CALMIN()

Description

The CALMIN() function corresponds to the minute of the calendar date you defined as part of the warm-up period or simulation begin date under simulation options. Since this function ties directly to the 24-hour clock displayed during simulation, CALMIN() will never return a value higher than 59.

Valid In

Any logic.

Please note

CALMIN() works only when you select calendar date in the simulation options dialog.

Example

Suppose parts have a fixed shipping time of 4:30 PM. The following logic would order parts to the shipping area.

```
IF CALHOUR() = 16 AND CALMIN() = 30 THEN

ORDER 1 Part TO Shipping_Area
```

See Also

CALDAY(), CALDOM(), CALHOUR(), CALMONTH(), and CALYEAR().

CalMonth()

General System Function

Syntax samples

```
CALMONTH()
```

Description

The CALMONTH() function corresponds to the month of the year you defined as part of the warm-up period or simulation begin date under simulation options. Values returned by this function will be integers in the range of 1 to 12.

Valid In

Any logic.

Please note

CALMONTH() works only when you select calendar date in the simulation options dialog.

Example

```
If CalMonth()=12 THEN WAIT 20
Else Wait 10
```

See Also

CALDAY(), CALDOM(), CALHOUR(), CALMIN(), and CALYEAR().

CalYear()

General System Function

Syntax samples

```
CALYEAR()
```

Description

The CALYEAR() function corresponds to the year of the calendar date you defined as part of the warm-up period or simulation begin date under simulation options.

Valid In

Any logic.

Please note

CALYEAR() works only when you select calendar date in the simulation options dialog.

See Also

CALDAY(), CALDOM(), CALHOUR(), CALMIN(), and CALMONTH().

Cap()

General System Function

Syntax samples

CAP (<*location*>)

GROUP CAP(Loc1)

Description

Returns the total capacity of a location. CAP() can be used to prepare a batch of entities to fill a location.

Valid In

Any logic and any expression evaluated after translation. For a list of expressions evaluated after translation, see the “Appendix A” on page 587.

Components

<location>

The name of the location to examine. If this component is the name of a multi-unit location, CAP() will return the capacity of the entire location. For example, if each unit of a five-unit location has a capacity of five, CAP() will return twenty-five. This component can also be the name of an individual unit in a multi-unit location.

Example

The individual ovens of a ceramics plant have different capacities, but otherwise they are exactly the same. While each oven could be modeled as an individual location, it would be easier to model this situation as a multi-unit location. This example uses CAP() in the processing logic of the parent location to accumulate enough Sinks to fill each oven. The LOCATION() function returns the value of the current location.

Process Table

	Location	Operation (min)
Sink	Oven	ACCUM CAP(LOCATION())

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Sink	Cool	FIRST 1	MOVE FOR 5

See Also

FREECAP(), LOC(), and LOCATION().

Char()

String Function

Syntax samples

CHAR(<*expression*>)

CHAR(10)

Description

Returns the ASCII character for the number that the expression evaluates to. This function is most useful for outputting ASCII characters that cannot be typed, such as accented characters. This function is most often used in conjunction with a string expression and the concatenation ("$") operator.

The number corresponding to the ASCII character varies with each computer depending on the character set specified in the config.sys file of your computer. To determine the number corresponding to the ASCII character at your computer, run the model CHAR.MOD found in the MODELS\REFS directory. View the file this model generates called CHAR.TXT found in the same directory as the model.

Valid In

Any string expression.

Components

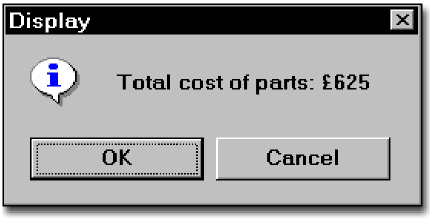
<expression>

Any expression that evaluates to a number between 0 and 255. The expression is evaluated every time the CHAR() function is encountered.

Example

The logic below displays the combined cost of all the Parts ordered through a particular service center. The simulation has kept a tally of all the parts it has ordered in the variable Parts, and each part cost 25 English pounds. The logic displays the cost of the parts with the pound symbol (£), which is ASCII code 156.

DISPLAY "Total cost of parts:" $ CHAR(156) $ Parts * 25



Total cost of parts: £625

See Also

"String Expressions" on page 412.

Clock()

General System Function

Syntax samples

CLOCK({<*time unit*>})

IF CLOCK(DAY) >= 1.5 THEN PAUSE

Attr1 = CLOCK()

Description

Returns value of the elapsed simulation time in the units specified. Clock units should be kept consistent when comparing values. If an attribute has been assigned a time in minutes, any time value compared with that attribute should also be in minutes.

When no units are specified in the parentheses in the clock function, ProModel returns the default time unit specified in the General Information.

Valid In

Any logic and most fields, except those fields evaluated at translation. For a list of fields evaluated at translation, see the "Appendix A" on page 587.

Components

<time unit>

The elapsed simulation time will be returned in this unit. This may be HR, MIN, SEC, DAY, WK. If omitted, the value returned will be in the default time units as specified in the General Information dialog box. When using CLOCK() to capture a time for later use with the LOG statement, the units should be omitted.

See Also

"General Information" on page 179 and "Time Expressions" on page 411.

Close

General Operation Statement

Syntax samples

```
CLOSE <file ID>
CLOSE Arrival_File
CLOSE ALL
```

Description

Closes a file that has previously been written to with WRITE, WRITELINE, XWRITE, or read with READ. Use CLOSE when finished with a file to free system resources. A file will automatically be re-opened to be read or written if it has been closed. All opened files are automatically closed at the end of a simulation. When you are using many external files and you want to conserve system resources, this statement is especially helpful.

Valid In

Any logic.

Components

<file ID>

The file ID of the desired file as defined in the External Files editor.

See Also

READ, WRITE, WRITELINE, XWRITE, and RESET.

Combine

Entity-Related Operation Statement

Syntax samples

COMBINE <*expression*> {AS <*new entity name*>}

COMBINE Var1

COMBINE 3 AS EntQ

COMBINE Var1 as ENT(Attr1)

Description

Accumulates and consolidates a specified quantity of entities into an entity, optionally with a different name. Unlike the GROUP statement, combined entities lose their identities and attributes and cannot be ungrouped later. Use COMBINE when several entities need to be combined, such as when eight spark plugs are combined in a box. Note that after several entities have been combined at a location, no additional statistics will be collected for any of the combined entities at that location.

When specifying COMBINE <expression> AS <new entity name> in the operation logic, there must be another operating block at the same location. In this case, the incoming entity at the new operating block is the new entity name specified in the COMBINE statement.

Valid In

The operation column of Process edit tables only. COMBINE may not be used in combination with CREATE, GROUP, UNGROUP, LOAD, UNLOAD, SPLIT AS, or other combine statements in the same process logic.

Components

<expression>

The number of entities to combine. Negative values generate an error message. If this expression evaluates to zero, it is ignored. If it evaluates to one, then no entities are actually combined, but the entity that encountered the combine statement is renamed (if the AS option has been specified).

This expression is evaluated every time an entity encounters the COMBINE statement, so the amount of entities to be combined can vary as the simulation progresses. If an entity arrives that changes this expression to a number lower than the number of entities already waiting to be combined, all of the entities waiting to be combined are combined, including the entity that just arrived.

AS <new entity name>

The optional name of the resulting entity. If left off, the new entity will have the same name as the last combined entity or the output entity name.

Explicit Entity Actions

COMBINE passes cost on to new entities but not statistical information. ProModel counts combined entities as exits.

Example

A manufacturing plant makes computer motherboards. After manufacture and inspection, when the motherboards are ready for shipping, workers combine them into sets of twenty to distribute to the company's customers. A COMBINE statement will work well for this example, because workers do not inspect or use the individual motherboards again. At the final assembly location, workers group motherboards into totes of twenty, and route the totes to the shipping department.

Process Table

	Location	Operation (min)
mboard	Assembly	COMBINE 20

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Tote	Shipping	FIRST 1	MOVE FOR 5

See Also

GROUP, ENT(), ACCUM, and LOAD. Also see the "Attributes" on page 225.

Comments

Documentation Symbols

Syntax samples

```
#
//
/*...*/
```

Components

#

The pound sign signals the start of a one-line comment. ProModel will ignore any characters on the rest of the line.

//

Two forward slashes signal the start of a one-line comment. ProModel will ignore any characters on the rest of the line. This symbol works exactly the same as the # sign.

/*...*/

A slash followed by an asterisk signals the start of a multi-line comment. ProModel will ignore all characters after the "/*" until it finds an asterisk followed by a slash, "*/". Use this type of comment for long explanations and to prevent ProModel from executing long portions of logic during debugging. Comments using // or # may be nested inside multi-line comments.

Description

Comments are notes to the modeler inside blocks of logic. ProModel ignores them, but they can be particularly useful to explain logic when more than one person will be using a model.

Valid In

Any logic.

Example

The logic below has several notes to explain it. Additionally, ProModel ignores the ELSE statement and the ELSE statement's block.

```
IF Parts_Available > Attr2 THEN
//Parts needed is stored in attribute 2
BEGIN
JOIN Attr2#Join the number of parts
# needed only if there are
# enough parts available.
WAIT Attr1
END
/* ELSE
BEGIN//Start operation
INC Var1
WAIT Attr3
END */
```

Contents()

General System Function

Syntax samples

CONTENTS(<*location*>{,<*entity type*>})

LOAD CONTENTS(Loc1)

JOIN CONTENTS(Loc1, EntA) EntA

Description

Returns either the total number of entities at a location or the number of a certain type of entity at a location. Use CONTENTS() to make decisions based on how busy a location is. Using CONTENTS() to keep track of the number of entities at a location requires fewer steps and allows more flexibility than using variables to count the number of entities that enter and exit a location. For example, the second syntax does in one statement what would require several statements without the CONTENTS() function.

Valid In

Any logic and all fields except those fields evaluated at translation only. For a list of the fields evaluated only at translation, see the "Appendix A" on page 587.

Components

<location>

The name of the location to examine.

<entity type>

The optional name of the type of entity to look for. If omitted, CONTENTS() will return the total of all entities at the location.

Example

An assembly line has a location called Clean that often gets too busy for one operator to handle and the supervisor then comes to help. The processing logic below models this situation with an IF...THEN statement and the CONTENTS() function. As long as the location contains fewer than five entities, a worker processes any arriving entities. However, if the location's contents are greater than five, the Supervisor processes them.

Process Table

	Location	Operation (min)

ALL	Clean	IF CONTENTS(Clean)<5
		THEN
		  USE Worker FOR 10
		ELSE
		  USE Worker FOR 10
		  OR Supervisor FOR 10

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	ALL	Mold	FIRST 1	MOVE FOR 1

See Also

FREECAP(), LOC(), ENT(), and FREEUNITS().

Create

Entity-Related Operation Statement

Syntax samples

```
CREATE <expression1> {AS <entity name>}
    {TAKE {<expression2>} <resource>,...}
CREATE 10 AS EntX
CREATE 2 AS EntB TAKE 1 Res1, 2 Res2
CREATE Var1 AS Ent(Var2) TAKE Var3
Res(Var4)
```

Description

Creates a specified number of entities in addition to the original entity and copies all attributes of the original entity to each new entity. The first entity created can optionally take any of the resources owned by the parent entity. The newly created entities require no additional capacity at the location and are processed immediately.

The CREATE statement can simulate the creation of paperwork that needs to be hand-carried to another location to be approved while the base entity continues to process. Before the base entity can exit the location, the paperwork must be approved and routed back to the original location where it is joined to the base entity.

Valid In

The operation column of process edit tables only. CREATE may not be used in combination with LOAD, UNLOAD, GROUP, UNGROUP, SPLIT AS, or other CREATE statements in the same processing logic.

Components

<expression1>

The number of new entities to create. This expression is evaluated every time the create statement is encountered.

AS <entity name>

The name of the new entities. ProModel will search the process list for the new entity type and begin processing them before it finishes processing the entity that created them. If omitted, the entities will be named the same as the currently processing entity.

TAKE <expression2>

The *first* created entity will take any resources listed here from the parent entity. This component is optional. The second syntax example above creates two EntB's. The first of the two will own one unit of Res1 and two units of Res2.

<resource>

The name of the resource whose ownership should be transferred to the first new entity. Using the keyword ALL here will take all resources owned by the parent entity.

Explicit Entity Actions

The CREATE statement forms a new entity with new statistical information and cost. ProModel adds an initial cost to *explicitly* created entities (i.e., entities created with the CREATE statement).

Implicit Entity Actions

ProModel allows you to use the CREATE statement *implicitly* as part of the routing definition. To do this, define a route block and check the New Entity option. ProModel does NOT add an initial cost to *implicitly* created entities.

Example

The following example shows how one entity, Record, creates two new entities. Note that there is no routing defined in this process for the new entities, Copy. The new entities are handled according to the logic defined in the subsequent process.

Process Table

	Location	Operation (min)
Record	Station1	WAIT N(8,.3) CREATE 2 AS Copy
Copy	Station1	WAIT U(3,.3)

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Record	Station2	FIRST 1	MOVE FOR 5
1	Copy	File_Cab	FIRST 1	MOVE FOR 5

See Also

ORDER, SPLIT AS, ENT(), and RES().

Debug

General Action Statement

Syntax samples

DEBUG

Description

Brings up ProModel’s debugger. Use DEBUG to step through logic one statement at a time and examine variable and attribute values while developing a model. After a model is working, DEBUG statements are generally removed.

Valid In

Any logic.

Example

If you were having trouble with a Client’s logic at a particular location, we could start the processing logic with a DEBUG statement, as in the following example. This would allow you to watch each Client’s logic execute, statement by statement, revealing problems with the logic’s flow.

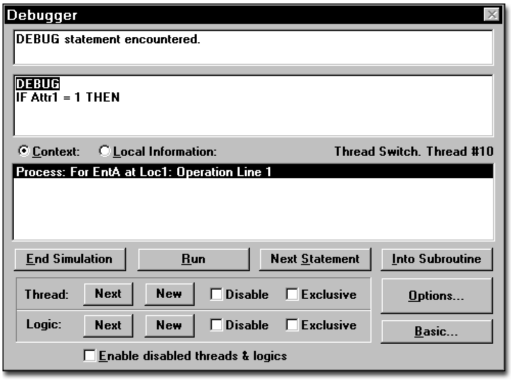
Process Table

	Location	Operation (min)
Client	Recep	DEBUG IF Attr1=1 THEN GRAPHIC WAIT 5 min

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Client	Waiting	FIRST 1	MOVE FOR 5

During run time, the ProModel debugger would appear displaying the following information:



DEBUG statement encountered.

```
DEBUG
IF Attr1 = 1 THEN
```

Thread Switch. Thread #10

Process: For EntA at Loc1: Operation Line 1

See Also

“Debug Option” on page 357.

Dec

General Operation Statement

Syntax samples

DEC <name>{, <expression>}

DEC Var1

DEC Attr1, 5

Description

Decrements a variable, array element, or attribute by the value of a specified numeric expression. To decrement a variable, attribute, or array element when the current entity actually leaves a location, use DEC in the move logic.

Valid In

Any logic.

Components

<name>

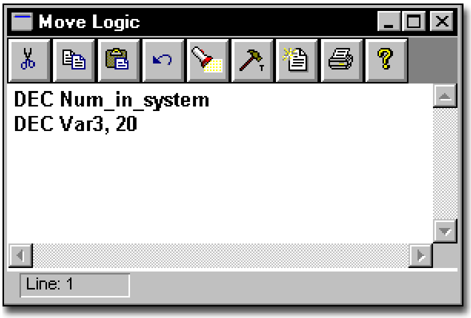
Any variable, array element, or attribute.

<expression>

The amount to decrement the value. If this expression is omitted, the value will be decremented by one. This can be a negative number.

Example

The example below shows two variables decremented in exit logic. The variable Num_in_system is decremented by one, while variable Var3 is decremented by 20.



```
DEC Num_in_system
DEC Var3, 20
```

See Also

INC. Also see “Numeric Expressions” on page 409.

Display

General Action Statement

Syntax samples

DISPLAY <string expression>

DISPLAY "Var1 =" $ Var1 $ "and Attr1 =" $ Attr1

DISPLAY "Now beginning 100th process"

DISPLAY Number_in_Queue

Description

Pauses the simulation and displays a message. The simulation will resume when the user selects OK. The concatenation operator ($) should be used to combine a numeric value into the string (as in the first syntax example above). Using the ENT(), LOC(), and RES() functions will display the name of the entity, location, or resource.

Valid In

Any logic.

Components

<string expression>

The message to be displayed. To display a numeric value, use the concatenation operator ($) as in the first syntax example.

Example

This simple example displays the value of both Var1 and of Attr1 if Attr2 is 1. This logic will display the dialog box below if Attr2 is 1. If Attr2 is not 1, an error message will appear.

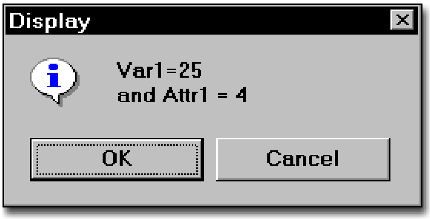
Operation (min)

IF Attr2=1 THEN

DISPLAY "Var1 =" $ Var1 $ "\nand Attr1 =" $ Attr1

ELSE

DISPLAY "Error"



Var1=25
and Attr1 = 4

Please note

The "\n" character starts new lines.

See Also

PROMPT, PAUSE, CHAR(), and FORMAT().

Do...Until

General Control Statement

Syntax samples

DO <statement block> UNTIL <Boolean expression>

```
DO INC Var1 UNTIL Array1(Var1) <> 10
```

```
DO
    BEGIN
        INC Var2, 5
        WAIT 5 sec
    END
UNTIL FreeCap(Loc1) > 5
```

Description

Repeats a statement or statement block continuously while a condition remains false. DO...UNTIL is an exit-condition loop, meaning that the loop will always be executed at least once. Use DO...UNTIL when an operation will always be executed at least one time and possibly more times.

Valid In

Any logic.

Components

<statement block>

The statement or block of statements to execute.

<Boolean expression>

As long as this expression is FALSE, the loop will continue. This expression is evaluated for each iteration of the loop.

Example

A machining station can manufacture parts of increasing complexity from the same entity, called a Blank. When a Blank arrives at the station, the value stored in Attr1 determines the complexity of the part and the amount of time needed to create the new part. The following logic models this situation with a DO...UNTIL loop. All blanks that arrive go through a five minute processing time, and then go through the operation several more times depending on the value of Attr1.

Process Table

	Location	Operation (min)

Blank	Machining	INT Count = 0
		DO
		  BEGIN
		    WAIT 5 min
		    INC Count
		  END
		UNTIL Count = Attr1

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Base	Painting	FIRST 1	

See Also

BEGIN, END, DO...WHILE, and WHILE...DO.

Do...While

General Control Statement

Syntax samples

DO <statement block> WHILE <Boolean expression>

```
DO INC Var1 WHILE Array1(Var1) <> 10

DO
    BEGIN
        INC Var2, 5
        WAIT 5 sec
    END
WHILE FreeCap(Loc1) > 5
```

Description

Repeats a statement or statement block continuously while a condition remains true. DO...WHILE is an exit-condition loop, meaning that the loop will always execute at least once. Use DO...WHILE for processes that must be executed one time and possibly more.

Please note

Use caution when using a DO...WHILE with a system function (e.g., FREECAP()) to ensure that the system does not enter an infinite loop. For example, eliminating the "WAIT 5 sec" in the syntax sample will cause the system to enter an infinite loop because there is no time delay within the loop.

Valid In

Any logic.

Components

<statement block>

The statement or block of statements to execute.

<Boolean expression>

As long as this expression is TRUE, the loop will continue. This expression is evaluated for each iteration of the loop.

Example

The logic below orders a minimum of ten cases to location Receiving every time it is encountered. As long as the location has a capacity greater than ten, it will send additional sets of ten entities to the location.

```
DO
ORDER 10 Cases to Receiving
WHILE FREECAP(Receiving) > 10
```

See Also

BEGIN, END, DO...UNTIL, and WHILE...DO.

Down

Downtime-Specific System Function

Syntax samples

```
DOWN <dtname>, {<priority>}
```

Description

Makes a location, with the specified called downtime, attempt to go down.

This statement is used in conjunction with the Called Downtime dialog, where you have previously defined the called downtime's name, priority, and logic.

When this statement is executed, the called downtime will attempt to execute its logic. The timing of the execution of the called downtime's logic will depend on the location's state and the called downtime's priority.

Valid In

Any logic, except Initialization and Termination logic.

Components

<dtname>

The name of the called downtime. This name is defined in the Called Downtime dialog, found in the Locations table.

<priority>

You may optionally define a priority. This will override the priority you defined in the Called Downtime dialog for the specified called downtime.

Please Note

The DOWN statement does not need to be called from the processing logic at the location that is to go down. It can be called from any logic in your model, except Initialization and Termination logic, and still cause the location's Called Downtime to execute.

See Also

"Called Downtime Editor" on page 111, and "Location Priorities and Preemption" on page 111.

DownQty()

General System Function

Syntax samples

DOWNQTY(<location> or <resource>>)

IF DOWNQTY(Loc1) > 3 THEN ROUTE 2

DISPLAY "Total Res1 Down Now:" $ DOWNQTY(Res1)

Description

Returns the number of location or resource units down at the time of the call. Use this function to make decisions based on how many resource or location units are down. For example, if too many units are down, a foreman could preempt several units back into service.

Valid In

Any logic and all fields except those fields evaluated at translation only. For a list of the fields evaluated only at translation, see the "Appendix A" on page 587.

Components

<location>

The name of the location to examine. LOC() can be substituted for the name of a location.

<resource>

The name of the resource to examine. RES() can be substituted for the name of a resource.

Example

Two resources, Welders, weld brackets onto steel frames at a location, Man_Weld, in 6.5 minutes. An automatic welding machine, Auto_Weld, can perform the same task, but it takes 9.3 minutes. However, if only one Welder is available and the other Welder is down (e.g., on break), it takes one Welder 13.0 minutes to weld the brackets to the frames. Therefore, if one Welder is down, the frames should route to the automatic welding machine, Auto_Weld.

Process Table

	Location	Operation (min)

Frame	Buffer	IF DOWNQTY(Welder)>0
		THEN
		ROUTE 1
		ELSE
		ROUTE 2

Frame	Auto_Weld	...
Frame	Man_Weld	...

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Frame	Auto_Weld	FIRST 1	MOVE FOR 1
2	Frame	Man_Weld	FIRST 1	MOVE FOR 1
...	...	...	...	...
...	...	...	...	...

See Also

FREECAP, FREEUNITS(), and UNITS().

DTDelay()

Downtime-Specific System Function

Syntax samples

DTDELAY(<*time unit*>)

WAIT Att1 - DTDELAY(Min)

DISPLAY "The downtime delay was"$DTDelay(DAY)$"days."

Description

Returns the difference between the time when you scheduled a non-preemptive downtime to occur and the time it actually occurred. Use DTDelay to determine if downtimes are being postponed because of incompleted work. You can use DTDelay in downtime logic to ensure that a location comes back up at a specific time.

Also returns the difference between the time when a downtime is preempted and the time it resumes.

Valid In

Any downtime logic including off-shift and break logic. This function returns a real number.

Components

<time unit>

The function will return the downtime delay in any of the following units: SEC, MIN, HR, DAY, and WK.

Example

The following statement models a situation where a location is supposed to go down at 12:00 and always goes back up at 1:00, even if it stays up past 12:00 to finish processing an entity. If the downtime was missed entirely (meaning that the downtime delay was greater than an hour), then the downtime takes no time at all. If the downtime was not missed entirely, then it lasts one hour minus the delay.

IF 60 - DTDelay(min) > 0 THEN WAIT (60 - DTDelay(min))

See Also

DOWNQTY().

DTLeft()

Shift & Break System Function

Syntax samples

```
DTLEFT <time units>
Shift_Time = DTLeft()
```

Description

The DTLEFT function returns the balance of the off-shift or break time remaining for a location or resource before it resumes normal activities. Unless passed as an argument (e.g., DTLeft(hr)), the return value uses the time units defined in the General Information Dialog (see “General Information” on page 179).

Valid In

This function can only be referenced in off-shift and break logic (either the pre-start or main logic).

Components

<time units>

The time units, if different from the default found in the General Information dialog, in which you want the return value expressed.

Example

Suppose a technician is frequently working on a job when the end of the shift rolls around and that the priority for ending the shift is not high enough to interrupt the job. To ensure that the technician gets a minimum of 14 hours off before coming back to work, even though the shift file specified 16 hours between shifts, you could enter the following off-shift logic:

```
IF DTLEFT(hr) < 14 THEN
BEGIN
WAIT 14 hr
SKIP
END
```

Please note

In the above example, the SKIP statement is important to skip over the defined time in the shift file.

See Also

DTDELAY()

DynPlot()

General Action Statement

Syntax samples

DYNPLOT "<*chart name*>"
DYNPLOT "my dynamic plot name"
DYNPLOT "Chart 1"
DYNPLOT ""

Description

The DynPlot statement is used to automatically turn on predefined Dynamic Plot windows. Its usage is similar to that of the View statement. Dynamic Plot windows are predefined during simulation mode. These predefined Chart views can then be called from any logic statement, invoking the display of the designated chart during simulation.

Valid In

Any Logic.

Components

<Chart name>

The name of the Dynamic Plot chart as defined in the Dynamic Plots dialog. Enclose the name in quotation marks.

Example

You are giving a presentation to management. Two hours into the simulation, you want a chart to be displayed showing the value of a WIP variable plotted against average part cost. Four hours into the simulation, you want this chart to disappear.

To do this, define a Chart view from the Dynamic Plots dialog, naming your Chart view "WIP vs Cost." Define a subroutine and call it in the initialization logic using the ACTIVE statement. Enter the following logic in the subroutine:

```
WAIT 2 HR

DYNPLOT "Wip vs Cost"

Wait 2 HR

DYNPLOT ""
```

Please note

Using the statement with a null value (no name defined between the quotation marks) will close all currently open Dynamic Plot windows.

End

General Control Statement

Syntax samples

```
END or }

WHILE FREECAP(Loc1) > 5 DO
    BEGIN
        INC Var2, 5
        WAIT 5 sec
    END
```

Description

Defines a statement block along with a corresponding BEGIN statement. BEGIN and END are almost always used in conjunction with other control statements such as IF...THEN and DO...WHILE. Every END must pair with a BEGIN.

Valid In

Any logic.

Example

Compare the following examples:

The example below includes a BEGIN and END statement in the form of the "{" and "}" symbols. In this logic, if the attribute Attr1 equals one, ten Cases are ordered and the variable, Var1, increments. ProModel executes the statements within the BEGIN and END only if Attr1 equals one.

```
IF Attr1 = 1 THEN
{
ORDER 10 Case
INC Var1
}
```

Just as in the logic above, if Attr1 in the following example equals one, ten Cases are ordered. However, Var1 increments no matter what the value of Attr1. Without a statement block, the IF...THEN applies only to the very next statement, no matter how you format the logic.

```
IF Attr1 = 1 THEN
ORDER 10 Case
INC Var1
```

See Also

BEGIN, IF...THEN, DO...WHILE, WHILE...DO, DO...UNTIL, and END.

Ent()

Type Conversion Function

Syntax samples

```
ENT(<entity name-index number>)
SEND 10 ENT(Var1) TO Loc1
DISPLAY "Ent A has been combined with" $
ENT(Var1)
```

Description

Converts a name-index number or integer to an entity name. Use this function when a statement or function needs the name of an entity whose name index number is stored in an attribute, variable, or some other expression. ENT() can also be used to vary the entity that a statement references by using an expression for the name-index number. When used in an expression expecting a string, such as in the second syntax example above, ProModel will convert the name-index number to the actual name of the entity.

Valid In

Any logic where an entity name is normally used.

Components

<entity name-index number>

The name-index number of the entity desired. This component may be an expression which allows the referenced entity to change as the simulation progresses. Real numbers will be treated as integers.

Example

The logic below orders three batches of five different entity types to a location, Receiving.

```
Var1 = 1
WHILE Var1 <= 3 DO
    BEGIN
        ORDER 5 ENT(Var1) TO Receiving
        INC Var1
    END
```

See Also

LOC(), RES(), ENTITY(), and LOCATION().

Entity()

Entity-Specific System Function

Syntax samples

ENTITY({<expression>})

ENTITY()

ENTITY(Var1)

Description

Returns the name-index number of the current entity or a particular entity in an entity group. This function is especially useful in macros and subroutines that vary depending on which entity calls them. Use ENTITY() to determine what type of entity is processing when an entity type of ALL is specified at a location. For example, if a common area handles several different parts with essentially the same process, use an IF...THEN statement in conjunction with an ENTITY() to have individual statement blocks handle the details of the operation. This function returns an integer.

Valid In

Speed fields, traveling time fields, conveyor speed fields, Resource fields, operation logic, routing fields, arrival logic, debug user-condition field.

Components

<expression>

The number of the entity in a group to inspect. For example, ENTITY(3) returns the name-index number of the third entity in the group. If this option is omitted, the name-index number of the calling entity is returned.

Example

All cars from the motor pool must be washed after use. All cars except vans require ten minutes to wash, and vans require an additional ten minutes. The logic below models this situation.

Process Table

	Location	Operation (min)
ALL	Wash1	WAIT 10 min IF ENTITY() = Van THEN WAIT 10 min

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	ALL	Parking	FIRST 1	MOVE FOR 30

See Also

ENT() and LOCATION().

Entries()

General System Function

Syntax samples

ENTRIES(<*location*>)

DISPLAY "LocA has had" $ ENTRIES(LocA) $ "entries."

Description

Returns the total entries to a location. This function returns an integer.

Valid In

Any logic and any field except those evaluated only at translation. For a list of fields evaluated at translation see the "Appendix A" on page 587.

Components

<location>

The location to examine.

Example

A location, Stores, sends entities, Orders, to a location, Shipping. A variable, Batch_Time, tracks the amount of time it takes to process 50 Orders. After Loc1 processes 50 entities, we want to reset Batch_Time to zero.

Process Table

	Location	Operation (min)

Order	Stores	IF ENTRIES(Loc1) = 50
		THEN
		  Batch_Time = 0

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Order	Shipping	FIRST 1	MOVE FOR 30

See Also

IF...THEN...ELSE.

Exp()

Math Function

Syntax samples

```
EXP(<expression>)
Real1 = EXP(Real2)
```

Description

Returns the exponential of an expression. This function is equivalent to e^x.

Valid In

Any expression. This function returns a real number.

Components

<expression>

EXP() returns the exponential of this expression.

See Also

LN().

ForLocation()

Shift & Break System Function

Syntax samples

```
FORLOCATION()

IF FORLOCATION() THEN Priority 100

IF FORLOCATION() THEN
    INC Arr1 (1, 2)
ELSE
    INC Arr1 (2, 2)
```

Description

This function returns TRUE if the object executing the shift or break logic is a location.

Valid In

Shift or break logic.

Example

We assign three resources (Oper1, Oper2, and Oper3) and two locations (Mill1 and Mill2) to a shift. When the locations go off-shift, the example uses a resource called Maint for 10 minutes to clean up around the machines. Because we assigned the same shift to locations and resources, we need to check if the object executing the off-shift logic is a location. We place the following logic in the off-shift logic:

```
IF FORLOCATION() THEN USE Maint FOR 10 min
```

See Also

FORRESOURCE(). Also see “Shift & Break Logic” on page 305.

Format()

String Function

Syntax samples

FORMAT(<expression>, <total number of characters in expression> {,<digits after decimal>})

DISPLAY "The value of Var1 is" $ FORMAT(Var1, 5)

Description

Converts a number to a string in the format specified. FORMAT() should most often be used with the concatenation operator ("$") and another string, as in the syntax example above. Format is often used in conjunction with the XWRITE statement to produce formatted output.

Valid In

Any string expression.

Components

<expression>

This expression will be evaluated and converted to a string. If the expression results in an integer, it will be converted to a real.

<total number of characters in expression>

This expression formats the number so that it occupies a total space equal to the number of digits before + number of digits after the decimal + one character for the decimal point. For example if you were to do the following logic: XWRITE file1 Format (10.0 4 1) XWRITE file1 Format (1.0 4 1), it will show up in the file as 10.0   1.0 with a space before the 1.0.

<digits after decimal>

An expression that evaluates to the maximum number of digits to the right of the decimal. If there are more digits to the right of the decimal than this number, the excess digits will be truncated. If there are fewer digits, ProModel will pad the number with zeros.

Example

The logic below writes formatted output to a file with XWRITE and FORMAT.

XWRITE File1, "The variable Var1 is" $ FORMAT(Var1,5,2)

In this example, if the value of Var1 is 378.87654, it would be written to the file as:

The variable Var1 is 378.87

↑___ (Two spaces)

See Also

WRITE and WRITELINE.

ForResource()

Shift & Break System Function

Syntax samples

```
FORRESOURCE()
IF FORRESOURCE() THEN GET Res1
```

Description

This function returns TRUE if the object executing the shift or break logic is a resource.

Valid In

All Shift or Break logic.

Example

A shift called DAYS.SFT has five resources and eight locations assigned to it. When the resources go off-shift, it is desired to write the resource name and simulation time to a file called RES_TIME. The following logic is placed in the off-shift logic.

```
IF FORRESOURCE() THEN
BEGIN
XWRITE res_time, “Shift for” $ RES(Resource()) $
“ended at” $ Clock(hr) $ “.”
END
```

See Also

FORLOCATION(). Also see “Shift & Break Logic” on page 305.

Free

Resource-Related Operation Statement

Syntax samples

FREE {<quantity>} <resource>, {{quantity} <resource>...}

FREE Res1, 2 Res2, 5 Res3A

FREE ALL

FREE RES(Attr1)

Description

Frees resources which are currently “owned” by the current entity. These resources must have been captured through a GET or JOINTLY GET statement.

Valid In

Location processing logic and downtime logic.

Components

<quantity>

The number of the following resource to free. A value of zero is ignored and values less than zero result in an error. This quantity may be any numeric expression and is evaluated and truncated to an integer every time the FREE statement is encountered.

<resource>

The name of the resource to free. The ALL keyword may be used here to free all resources owned by an entity. Any resource which is not owned by the entity will be ignored.

Example

In the following example, EntA arrives at Loc1 for a multi-step process requiring the use of resources Res1 and Res2. The first step requires the simultaneous use of Res1 and Res2 for a normally distributed amount of time. ProModel then frees Res1while Res2 performs the second step of the process according to a Lognormal distribution.

Process Table

	Location	Operation (min)
EntA	Loc1	JOINTLY GET Res1 AND Res2 WAIT N(4.5,.2) FREE Res1 WAIT L(3.4,.23) FREE Res2

Routing Table

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

See Also

GET and JOINTLY GET.

FreeCap()

General System Function

Syntax samples

FREECAP(<*location*>)

SEND FREECAP(Loc1) EntA TO Loc1

Description

Returns the available capacity of a location. This function returns an integer.

Valid In

Any logic and any field except those fields evaluated only at translation time. For a list of fields evaluated only at translation time, see the "Appendix A" on page 587.

Components

<location>

The name of the location to examine. The LOC() function may also be used here.

Example

Suppose the entities, Plates and Cams, travel through an assembly line. The location, Assembly (capacity=1) joins Cams with the Plates. When a Cam finishes processing at Station2, it should not enter Station3 unless a Plate is waiting to join with it further down the assembly line at Assembly. If there is no Plate at Assembly, another location, Buffer, sends one. The logic for Cam at Station2 is as follows:

```
IF FREECAP(Assembly) = 1 THEN
SEND 1 Plate TO Assembly
```

See Also

CAP().

FreeUnits()

General System Function

Syntax samples

FREEUNITS(<*location*> or <*resource*>)

USE (FREEUNITS(Res1)) Res1 FOR 5 min

Description

Returns the free units of a location or resource.

Valid In

Any logic and any field except those fields evaluated only at translation time. For a list of fields evaluated only at translation time, see the “Appendix A” on page 587.

Components

<location>

The name of the location to examine.

<resource>

The name of the resource to examine.

Example

This example uses FREEUNITS() to assign a team of Specialists to rework a problem. The size of the team, stored in the local variable T_Size, is determined by all the free Specialist units. Team_Time is a table function that varies the amount of time it takes to solve a problem (the amount of time in the USE statement) based on the number of units on the team.

Process Table

	Location	Operation (min)
Problem	Rework	INT T_Size = FREUNITS(Specialist) IF T_Size = 0 THEN T_Size = 1 USE T_Size Specialist FOR Team_Time(T_Size) Hr

Routing Table

Blk	Output	Destination	Rule	Move Logic

See Also

UNITS(), RES(), and LOC().

Get

Resource-Related Operation/Move Logic Statement

Syntax samples

GET {<quantity>} <resource> {,<priority>} {AND or OR {quantity} <resource> {,<priority>}...}

GET Res1

GET 3 Res1, 5

GET 2 Res1OR 3 Res2

GET Res1, 3 AND (Res2 OR Res3)

GET Res(Skill_required)

Description

Captures a specified number of resources as they become available. If the entity already possesses one of the requested resources, the entity will still try to capture an additional unit of that resource. When capturing multiple resources, each resource will be captured as soon as it becomes available until all resources are captured.

A resource captured with the GET statement at one location and then released with a FREE statement at another location will not be used to move the entity between locations unless it is also specified in a MOVE WITH statement in the routing move logic. Otherwise, it is the entity that is moving the resource from one location to the next and the resource will not be visible when moving with the entity.

Resources captured by a GET statement can only be preempted when the entity owning the resource is undergoing a WAIT or USE time except in move logic. If the resource is preempted during one of those times, the time will continue where it left off when the resource becomes available.

Every GET must have a corresponding FREE or an error occurs when the entity exits the system. If an entity owns one or more resources and is subsequently loaded onto or grouped with another entity, it cannot free the resource(s) until it is unloaded or ungrouped.

Valid In

Location processing, downtime, move, and shift logic. A GET statement cannot follow a move related statement in move logic.

Components

<quantity>

The number of resources to get. A value of zero is ignored and values less than zero return an error. This numeric expression is evaluated and truncated every time the GET statement is encountered.

<resource>

The name of the resource to GET. You can substitute RES() for the resource name.

<priority>

When multiple entities are waiting for the same resource to become available, the entity with the highest priority will capture the resource first. This expression should be a number between 0 and 999.

Example

To start a semi-automatic welding process, we need a static resource named "Operator." Midway through the process, the Operator tests the weld with a resource named "Tester," shared with other operators. After the test, the operator restarts the welder. The Operator stays with the welder until he or she completes the last part of the operation. After the welding process the operator moves the part to the buffing area, completes the buffing operation and then is freed.

Process Table

	Location	Operation (min)
Assy	Weld	USE Operator FOR 4 min GET Operator USE Tester FOR U(2,.5) WAIT 4 min
Assy	Buffer	WAIT T(7,10,11) FREE Operator

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Assy	Buffer	FIRST 1	MOVE FOR 1.5
1	Assy	Checker	FIRST 1	MOVE FOR 5

See Also

JOINTLY GET and USE.

GetCost()

Cost Related Function

Syntax samples

GETCOST()

Description

Returns the cost of the current entity executing the logic. Use this function to return the entity's actual, accumulated dollar amount.

Valid In

Operation logic only.

Example

The following example shows how to use the GETCOST() function to generate a Time Series plot that tracks changing entity cost as entities exit the system. (See *table below.*)

- Create a variable (e.g., Var1), select it to be of type real, and select time series statistics.
- For any location where an entity exits the system, place the following as the last line in operation logic:

Var1 = GETCOST()

Process Table

	Location	Opn (min)
EntA	Shipping	WAIT N(3,.52) Var1=GETCOST()

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Exit	FIRST 1	

See Also

INCENTCOST, INCLOCCOST, and SETRATE.

GetReplicationNum()

General System Function

Syntax samples

```
GETREPLICATIONNUM()
```

Description

Returns the number of the currently running replication.

Valid In

Any Logic.

Example

Based on the current replication, you may want to make a decision regarding the exporting of array data.

In this case, you could use an IF THEN statement combined with the GETREPLICATIONNUM() function to decide what data to export based on the currently running replication.

GetResRate()

Cost Related Function

Syntax samples

```
GETRESRATE({<resource>})
GETRESRATE()
GETRESRATE(Operator1)
```

Description

Returns the cost rate specified in the Cost dialog or through the SETRATE() function for a resource currently owned by the entity making the function call. When used without the optional <*resource*> parameter, this function returns the cost rate of the entity's most recently captured, owned resource.

If an entity owns multiple units of a resource, the function returns the cost rate of the entity's most recently captured resource unit.

Valid In

Operation and move logic.

Components

<resource>

A resource currently owned by the entity making the function call. When you use GETRESRATE without this parameter, this function returns the cost rate of the entity's most recently captured, owned resource.

Example

A clerk normally works an 8-hour shift. However, if the clerk does not finish entering all the orders in his Input_Queue at the end of the day, he must stay and finish them. Since the clerk is paid time-and-a-half for any overtime, you must increment the cost of the resource by an additional half of the normal rate to get the overtime rate. To do this, set a variable equal to 1 in the pre-off-shift logic for the resource, indicating the resource should go off-shift. If the variable is equal to 1, increment the cost of a resource by half the original rate. (Since each unit of the clerk has a different rate, you must obtain the cost rate for the resource owned by the entity.)

Process Table

	Location	Opn (min)
Order	Order_desk	GET Clerk, 399 WAIT N (4.5, .3) IF Off_Shift_Var=1 THEN INCRESCOST GETRESRATE() *.50 FREE Clerk

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Order	Shipping	FIRST 1	MOVE FOR 1

See Also

GETCOST(), INCENTCOST, INCLOCCOST, INCRESCOST, and SETRATE.

Goto

General Control Statement

Syntax samples

GOTO <*label ID*>

GOTO LabelA

Description

Jumps to the statement identified by the designated label. A label should follow the normal rules for names except that it is followed by a colon in the logic. GOTO statements may be replaced by IF...THEN...ELSE statements.

Valid In

Any logic.

Components

<label ID>

The name of the label to switch to. Omit the colon on the label name here.

Example

This example shows how a GOTO statement is used to skip over the first two increment statements based on the value of an attribute.

Process Table

	Location	Operation (min)
Box	Receive	IF Attr1>1 THEN GOTO L1 INC V1 INC V2 L1: INC V3

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Box	Stores	FIRST 1	MOVE FOR 5

See Also

IF...THEN...ELSE, BREAK, and BREAKBLK.

Graphic

General Operation Statement

Syntax samples

GRAPHIC <*expression*>

GRAPHIC 2

GRAPHIC Var1

Description

Changes the entity's or resource's current graphic. Entities and resources are assigned graphics from the graphics library in the Entity or Resource editor. Use the GRAPHIC to show the result of a process. For example, when a customer sits down, the graphic could change from a standing person to a sitting person.

Valid In

When used in node entry, node exit, and resource-downtime logic, GRAPHIC changes a resource's graphic. When used in location processing logic, move logic, and arrival logic, GRAPHIC changes an entity's graphic.

Components

<expression>

The number of the entity's or resource's new graphic.

Example

In the example below, EntA arrives at Loc1 for a two step operation. Upon arrival, the graphic icon of the entity changes to Graphic 2. After the first two minute processing time the icon changes to Graphic 3. Finally, the icon changes to Graphic 4 after the second two minute processing time. (The difference between the two graphics may only be color, but the two could be completely different.)

Process Table

	Location	Operation (min)
EntA	Loc1	GRAPHIC 2 WAIT 15 min GRAPHIC 3 WAIT 25 min GRAPHIC 4

Routing Table

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

See Also

"Entities" on page 118 and "Resources" on page 132.

Group

Entity-Related Operation Statement

Syntax samples

```
GROUP <expression> {AS <entity name>}
GROUP (Var1+Var2)
GROUP 10 AS EntX
```

Description

Accumulates and temporarily consolidates a specified quantity of entities into a single GROUP shell entity.

The shell entity retains the same attributes as the first entity that was grouped into the shell. However, if the GROUP AS statement is used, the new shell entity does not retain any attribute values, even if the same name is used for the GROUP shell entity as the entities that have been grouped.

The individual entities comprising the group retain their identities, attributes, and resources and are divided from the group when an UNGROUP statement is encountered. The first entity processed from the group takes any resources the group owns. Entities in a group can be grouped into a larger group at another location.

Valid In

The operation column of process edit tables only. GROUP may not be used in combination with COMBINE, CREATE, LOAD, UNLOAD, SPLIT AS, other GROUP statements in the same processing logic, or with conveyors. An exception to this rule is that an UNGROUP statement may follow a GROUP statement in the same process.

Components

<expression>

The number of entities to group. If this expression is zero, the statement is ignored. If it is less than zero, it generates an error.

This expression is evaluated every time an entity encounters the GROUP statement, so the amount of entities to be combined can vary as the simulation progresses. If it becomes less than or equal to the number of entities already waiting to be combined, the entity that encountered the GROUP statement will be grouped with all the waiting entities.

AS <new entity name>

The optional name of the resulting entity. If left off, the new entity will have the same name as the last entity to complete the group.

Explicit Entity Actions

GROUP creates a shell (a temporary entity representing grouped entities that starts with cost and time statistics of zero) to which ProModel assigns all cost and time statistics for the group. Each grouped entity retains its individual cost and time statistics and, when you ungroup the entities and the shell disappears, ProModel divides all statistics and cost (accrued by the shell) between them.

Example

In this example, Man, Woman, and Child are grouped together at Floor1 before getting on an elevator. The group of entity types, called Grp_A, is then routed to Floor2 where it will be ungrouped. (See the UNGROUP statement example.)

Process Table

	Location	Operation (min)
Man	Floor1	WAIT E(2.0)
Woman	Floor1	WAIT U(3,1)
Child	Floor1	WAIT N(2.1,.2)
ALL	Waiting	GROUP 10 AS Grp_A
Grp_A	Waiting	GET Elevator

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Man	Waiting	FIRST 1	MOVE FOR 0.5
1	Woman	Waiting	FIRST 1	MOVE FOR 0.5
1	Child	Waiting	FIRST 1	MOVE FOR 0.5
1	Grp_A	Floor2	FIRST 1	MOVE WITH Elevator

See Also

COMBINE, ACCUM, GROUPQTY(), and ENTITY(). Also see “Attributes” on page 225.

GroupQty()

Entity-Specific System Function

Syntax samples

GROUPQTY({<entity name>})

ORDER GROUPQTY(Part1) Part2 TO Loc1

IF GROUPQTY(Part1) > 5 THEN...

Description

Returns the number of entities of a specified type in a grouped or loaded entity. If no name is specified, it returns the entire group quantity. If it is a loaded entity, it will only return the number of loaded entities, not the base entity. For example, if four Castings are loaded onto a Pallet and called Batch, the GroupQty() will return the number of Castings (i.e., 4), which does not include the entity Pallet.

In the case of hybrid nested groups with several mixed and nested levels of groups and loads, GroupQty() returns the number of entities in the uppermost level only.

Valid In

Speed fields, traveling-time fields, conveyor-speed fields, Resource fields, operation logic, routing fields, arrival logic, and debug user-condition fields. This function returns an integer.

Components

<entity name>

The optional specific type of entity to search for in the group.

Example

A group of entities called GRPA arrives at Loc1 and process for some amount of time according to a Normal distribution. If the group contains three or more entities it routes to Loc2, otherwise it routes to Loc3. Routing requires 1.2 minutes.

Process Table

	Location	Operation (min)
GRPA	Loc1	WAIT N(3,.1) IF GROUPQTY() > 2 THEN ROUTE 1 ELSE ROUTE 2

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	GRPA	Loc2	FIRST 1	MOVE FOR 1.2
2	GRPA	Loc3	FIRST 1	MOVE FOR 1.2

See Also

GROUP, UNGROUP, and ENT().

If...Then...Else

General Control Statement

Syntax samples

```
IF <Boolean expression> THEN <statement 1>
{ELSE <statement 2>}


IF Var1 = 5 THEN WAIT 2 min


IF (Attr2 = 5) OR (Var5 <> 0) THEN WAIT 2 min
ELSE WAIT 3 min


IF Var1 > Attr2 THEN
    BEGIN
        Var1 = Attr2
        WAIT Attr1
    END
ELSE
    BEGIN
        INC Var1
        WAIT Attr2
    END
```

Description

Executes a statement or statement block if the Boolean expression is true. If an ELSE statement is included and the Boolean expression is false, an alternate statement or statement block is executed. For an IF...THEN statement to be broken into more than one line, the first item on the next line must be THEN, AND, or OR. IF...THEN statements only apply to the next statement or statement block in a logic. Any statements outside of the BEGIN and END will execute normally. See BEGIN and END for examples.

Valid In

Any logic.

Components

<Boolean expression>

The condition to be tested.

THEN <statement 1>

The statement or block to be executed if the condition is true.

ELSE <statement 2>

The statement or block to be executed if the condition is false.

Example

In the following example an IF...THEN...ELSE test is made to see if the simulation clock is less than eight hours. If so, the Client is routed to Office1, otherwise the Client is routed to Office2.

Process Table

	Location	Operation (min)

Client	Loc1	IF CLOCK(Hr) < 8 THEN
		ROUTE 1
		ELSE
		ROUTE 2

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Client	Office1	FIRST 1	MOVE FOR 5
2	Client	Office2	FIRST 1	MOVE FOR 5

See Also

BEGIN, END, DO...WHILE, WHILE...DO, and DO...UNTIL.

Inc

General Operation Statement

Syntax samples

INC <name>{, <expression>}

INC Var1

INC Attr2, 5+Var1

Description

Increments a variable, array element, or attribute by the value of a specified numeric expression. When counting the number of entities a location has processed, increment a variable at the end of the processing logic.

Valid In

Any logic.

Components

<name>

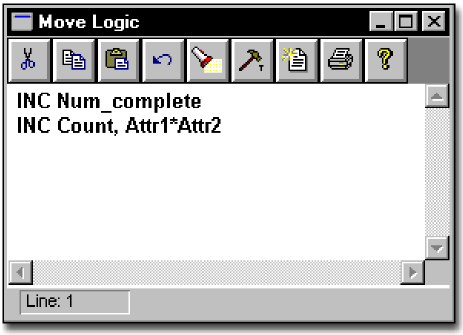
The name of any variable, array element, or attribute.

<expression>

The amount to increment the value. If this expression is omitted, the value will be incremented by one. This can be a negative number.

Example

The following example increments two variables in the exit logic. Num_complete is incremented by one, and Count is incremented by the expression Attr1*Attr2.



```
INC Num_complete
INC Count, Attr1*Attr2
```

See Also

DEC.

IncEntCost

Cost Related Statement

Syntax samples

INCENTCOST <expression>

INCENTCOST 15

INCENTCOST -15

Description

Enables you to increment the cost (positively or negatively) of the current entity by a given amount. Use this function to add to the entity's actual, accumulated dollar amount.

Please note

When working with new entities created through a ROUTE statement, use INCENTCOST to assign an initial cost to entities. Initial entity cost defined for an entity in the cost module applies to entities entering the system through a scheduled arrival, the CREATE statement, or the ORDER statement.

Valid In

Operation and Move logic.

Components

<expression>

The positive or negative change to the value of cost.

Example

The logic below allows you to add an initial cost to an entity implicitly created through the ROUTE statement. In the following example, a Pallet of entities, PalletA, arrives at Unload_Loc where workers unload entities called Box every 20 seconds until the pallet is empty. ProModel determines the number of Boxes unloaded from PalletA by the value of PalletA's attribute, Qty_Attr. In Move Logic, the statement "IncEntCost 10" adds an initial cost of 10 dollars to each implicitly created entity, Box.

Process Table

	Location	Opn (min)
PalletA	Unload_Loc	int x = 0 WHILE x < Qty_Attr DO {INC x WAIT 20 sec ROUTE 2} ROUTE 1

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	PalletA	Exit	FIRST 1	
2*	Box	Conveyor1	FIRST 1	INCENT-COST 10

See Also

GETCOST, GETRESRATE(), INCLOCCOST, INCRESCOST, and SETRATE.

IncLocCost

Cost Related Statement

Syntax samples

```
INCLOCCOST <expression>
INCLOCCOST 15
INCLOCCOST -15
```

Description

Enables you to increment the cost (positively or negatively) of the current location by a given amount. Use this function to add to the location's actual, accumulated dollar amount.

Valid In

Operation logic.

Components

<expression>

The positive or negative change to the value of cost.

Example

The logic below shows how to add a cost per entry to a location for an entering entity. ProModel automatically tracks operation cost per time unit specified in the cost dialog, however, you may assign a one time cost each time an entity enters a location.

In the following example, an entity, EntA, arrives for inspection at location Inspect. We define no rate of operation cost for Inspect because each inspection accrues a one time cost regardless of how long the entity remains at Inspect. At Inspect, workers inspect EntA. After inspection, ProModel adds the cost to the location through the IncLocCost statement and then to the entity with the IncEntCost statement.

Process Table

	Location	Opn (min)
EntA	Inspect	WAIT N(3,.7) INCLOCCOST 3 INCENTCOST 3

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Packaging	PROBABILITY .93	
	EntA	Rework	.07	

See Also

GETCOST, GETRESRATE(), INCENTCOST, INCRESCOST, and SETRATE.

IncResCost

Cost Related Statement

Syntax samples

INCRESCOST <cost expression> {,<resource>}

INCRESCOST 10

INCRESCOST GETRESRATE(Operator1)*20, Operator1

Description

Enables you to increment the cost (positively or negatively) of a resource currently owned by the entity executing the statement. Use this function to add to the resource's actual, accumulated dollar amount. When used without the optional *<resource>* parameter, this statement increments the cost rate of the entity's most recently captured, owned resource.

If an entity owns multiple units of a resource, the cost distributes evenly to each unit.

Valid In

Operation and move logic.

Components

<cost expression>

The positive or negative change to the value of cost.

<resource>

A resource currently owned by the entity executing the statement. When used without the parameter, this statement increments the cost rate of the entity's most recently captured, owned resource.

Example

A clerk normally works an 8-hour shift. However, if the clerk does not finish entering all the orders in his Input_Queue at the end of the day, he must stay and finish them. Since the clerk is paid time-and-a-half for any overtime, you must increment the cost of the resource by an additional half of the normal rate to get the overtime rate. To do this, set a variable equal to 1 in the pre-off-shift logic for the resource, indicating the resource should go off-shift. If the variable is equal to 1, increment the cost of a resource by half the original rate.

Process Table

	Location	Opn (min)
Order	Order_desk	GET Clerk, 399 WAIT N (4.5, .3) IF Off_Shift_Var=1 THEN INCRESCOST GETRESRATE() *.50 FREE Clerk

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Order	Shipping	FIRST 1	MOVE FOR 1

See Also

GETCOST(), GETRESRATE(), INCENTCOST, INCLOCCOST, INCRESCOST, and SETRATE.

Int

Local Variable Declaration Statement

Syntax samples

INT <name1>{= <expression1>, <name2>= <expression2>...)

INT Count

INT Count = 1

INT Count = 1, Test = FREECAP(Loc2)

Description

Creates a local variable of type integer. Local variables work much the same as attributes, except that they only are available within the logic that declares them. A new variable will be created for each entity that encounters an INT statement. Local variables are not directly available to subroutines, which have their own local variables. However, a local variable may be passed to a subroutine as a parameter. Local variables are available inside referenced macros.

Use local variables where ever possible for the test variable in WHILE...DO, DO...WHILE, and DO...UNTIL loops.

Valid In

Any logic. Variables declared with INT are valid in any expression within the logic where a normal integer number is valid.

Components

<names>

An identifier for the local variable. This identifier must be a valid name.

<expressions>

The variable will initially be assigned this value. This expression is evaluated every time the INT statement is encountered.

Example

A plant manufactures pipes of 12 different types, such as 2" steel, 4" aluminum, etc. All workers inspect pipes at a common inspection station after which they move to a dock where other workers load them into boxes. The boxes are designed to hold only certain types of pipes. Therefore a box designed to hold 2" steel pipes can only hold 2" steel pipes, not 4" aluminum pipes.

Suppose a Box, enters a multi-capacity location, Dock. Each Box has a different entity attribute, b_type, describing the type of pipe it can hold. Workers load pipes into the Box. Workers must load the 2" steel pipes into the box designed to hold the 2" steel pipes. Therefore, the attribute value of the Pipe, p_type, must match the attribute value of the Box, b_type. We can use local variables to accomplish this modeling task. In the following example, we defined X as a local variable and set it equal to b_type:

Process Table

	Location	Operation (min)
Pipe	Inspect	WAIT 5
Box	Dock	INT X= b_type LOAD 5 IFF X=p_type
Box	Dock	WAIT 10

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Pipe	Dock	LOAD 1	MOVE FOR 2
1	Box	Delivery	FIRST 1	MOVE FOR 8

See Also

REAL. Also see “Local Variables” on page 233.

Join

Entity-Related Operation Statement

Syntax samples

```
JOIN <expression> <entity name> {,<priority>}
JOIN 4 EntA
JOIN Var1 EntA, 1
```

Description

Joins a specified quantity of a designated entity type to the current entity. The entities joined to the current entity lose their identities and any resources owned by the joining entities are transferred automatically to the current entity. Use JOIN to simulate a component being assembled to a main part, such as when wings are attached to the fuselage of an airplane.

Entities to be joined must be routed to the current location with a JOIN rule. The current entity waits until enough entities to fill its request have been routed to the current location with a JOIN rule.

The resulting entity retains the attributes and name of the current entity. To transfer attributes from the joining entity to the current entity, in the exit logic for the joining entity, copy the desired attribute to a global variable. Then assign the global variable to the attribute of the current entity after the JOIN statement in the processing logic.

All resources owned by the joining entity are transferred to the base entity.

To JOIN an entity with a specific attribute value to another entity with the same attribute value, use the MATCH statement in addition to a JOIN.

Valid In

The operation column of process edit tables only. More than one JOIN statement may be used in the same logic.

Components

<expression>

The number of entities to be joined. A zero value is ignored and a negative value generates an error. This expression is evaluated when each entity first encounters it, but is not re-evaluated as the requested entities are joined.

<entity name>

The entity type to be joined to the processing entity. Joining entities must come from a JOIN routing and lose their identity once joined.

<priority>

An entity with a higher priority will have arriving entities joined to it before one with a lower priority. This expression should be a number between 0 and 999. For more information on priorities, see Priorities, at the beginning of this section.

Explicit Entity Actions

JOIN passes cost on to base but not statistical information. ProModel counts joined entities as exits.

Example

A certain location on an assembly line manufacturing cars attaches doors built on another assembly line. When the Body of the car arrives at the Assembly location, we attach a left door, Ldoor, and a right door, Rdoor, with a JOIN statement. The paint station paints the doors and routes them to Assembly with a JOIN rule. Note that the Body will wait until we join both doors before routing to the next location.

Process Table

	Location	Operation (min)
Rdoor	Paint	WAIT 30
Ldoor	Paint	WAIT 30
Body	Assembly	JOIN 1 Ldoor JOIN 1 Rdoor

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Rdoor	Assembly	JOIN 1	MOVE FOR 30
1	Ldoor	Assembly	JOIN 1	MOVE FOR 30
1	EntB	Cleaning	FIRST 1	MOVE FOR 5

See Also

LOAD, COMBINE, and GROUP. Also see “Attributes” on page 225.

Jointly Get

Resource-Related Operation Statement

Syntax samples

```
JOINTLY GET {<quantity>} <resource>
    {,<priority>}
    {AND or OR {<quantity>} <resource>
    {,<priority>}}
```

JOINTLY GET 3 Res1,5

JOINTLY GET 2 Res1 OR 3 Res2

JOINTLY GET Res1,3 AND (Res2 OR Res3)

JOINTLY GET 2 Res(Attribute1)

Description

Captures a specified number of resources when that number of resources is available. When capturing multiple resources, none of the resources will be captured until all are available. If the entity already possesses one or more of the requested resources, the entity will still try to capture additional units of the specified resources.

Valid In

Location processing, downtime, move, and shift logic.

Components

<quantity>

The number of resources to get. A value of zero is ignored and values less than zero return an error. This numeric expression is evaluated and truncated every time the JOINTLY GET statement is encountered.

<resource >

The name of the resource to JOINTLY GET. RES() can be substituted for the resource name.

<priority>

When multiple entities request a resource, the requests will be filled in order of priority. This expression should be a number between 0 and 999.

Example

In the following example (which also demonstrates the FREE statement) EntA arrives at Loc1 for a multi-step process. Because the first step of the process uses Res1 and Res2 simultaneously, a JOINTLY GET statement is issued to ensure that both resources are available before the process begins. The resources are then freed independently.

Process Table

	Location	Operation (min)
EntA	Loc1	JOINTLY GET Res1 AND Res 2 WAIT N(4.5,.2) FREE Res1 WAIT L(3.4,.23) FREE Res2

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	FIRST 1	MOVE FOR 1

See Also

GET and USE.

Last()

Resource-Specific System Function

Syntax samples

LAST()

Var1=LAST()

IF LAST() = 13 THEN Var3 = 0

IF LAST() = PathNet1.N1 THEN INC Var1

Description

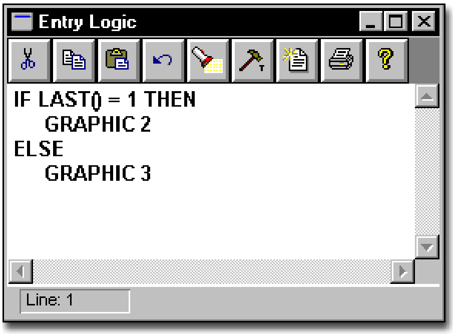
Returns the name-index number of the node from which a resource has just traveled. LAST() can be useful to choose the appropriate graphic or to reset a variable. You can also check the name-index number of the last node by specifying <path network name>. <name of the node>. For example, if you wanted to know if the last node was N5 on the network Net3, you could specify “IF LAST() = Net3.N5 THEN...” in the node entry logic.

Valid In

Node entry logic.

Example

This Entry Logic window shows that whenever a resource enters a particular node a check is made to see if the name-index number of the last node equals one. If so, the resource graphic is changed to Graphic 2, otherwise it is changed to Graphic 3.



```
IF LAST() = 1 THEN
   GRAPHIC 2
ELSE
   GRAPHIC 3
```

See Also

NEXT() and WAIT...UNTIL.

Ln()

Math Function

Syntax samples

LN(<*expression*>)

Real1 = LN(Real2)

Description

Returns the natural logarithm of an expression.

Valid In

Any expression. This function returns a real number.

Components

<expression>

LN() returns the natural logarithm of this expression.

See Also

EXP().

Please note

To get a logarithm to a particular base, use the following formula:

*logbase<expression> = LN<expression>/
LN<base>*

Load

Entity-Related Operation Statement

Syntax samples

LOAD <expression> {IFF <Boolean expression>}{IN <time>}{,<priority>}

LOAD 5, 99

LOAD 5 IFF Attr3 > 2 IN 5 min

LOAD Pallet_Capacity

Description

Loads a specified quantity of entities onto the current entity. Loaded entities retain their identity and may be unloaded with an UNLOAD statement. Loaded entities must be routed to the loading location using the LOAD routing rule. Additional entities may be added to an entity's existing load with additional LOAD statements. Use LOAD to model parts placed into a container or pallet when they must be removed later. If a resource owns the loaded entity when the entities are unloaded from the base entity, the resource stays with the base entity.

Valid In

The operation column of Process edit tables only. A process may contain multiple load statements and no more than one UNLOAD statement following all LOAD statements. LOAD may not be used in the same process with SPLIT AS, CREATE, COMBINE, GROUP, or UNGROUP.

Components

<expression>

The number of entities to load into the current entity. This expression is evaluated at the time the entity encounters the LOAD request.

IFF <Boolean expression>

This option allows the LOAD command to be conditional. Any attributes, entity functions, or location functions apply to the entity to be loaded, not to the current entity. This technique allows only entities with certain properties to be loaded onto the current entity. To use attributes, entity functions, and location functions that apply to the current entity, assign the desired value to a local variable and use the local variable in the Boolean expression.

IN <time>

The entity will load entities up to the specified limit for this amount of time and then go on. Entities loaded with this option may have a load smaller than the specified amount.

<priority>

Waiting entities will load arriving entities by priority. This expression should be a number between 0 and 999.

Explicit Entity Actions

LOAD does not transfer cost or statistics of the loaded entity.

Example

In this example, boxes are loaded onto a semi-truck. The quantity is determined by the value of the Truck's attribute, Attr1. The resulting entity retains the name Truck and is sent on to its final destination, New York, Chicago, or Boston.

Process Table

	Location	Operation (min)
Box	Shipping	WAIT 2 min
Truck	MfgSite	
Truck	Dock	LOAD Attr1 IN 2 hr

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Box	Dock	LOAD 1	MOVE FOR 45 sec
1	Truck	Dock	FIRST 1	MOVE FOR 10 min
1	Truck	NewYork	FIRST 1	MOVE FOR 24 hr
	Truck	Chicago	FIRST	MOVE FOR 12 hr
	Truck	Boston	FIRST	MOVE FOR 28 hr

See Also

LOAD, GROUP, UNLOAD, ACCUM, COMBINE, JOIN, GROUPQTY(), and ENTITY(). Also see “Attributes” on page 225.

Loc()

Name-Index-Number Conversion Function

Syntax samples

```
LOC(<location name-index number>)
ORDER 10 EntA TO LOC(5)
DISPLAY "EntA arrived at" $ LOC(5)
```

Description

Converts a name-index number or integer to a location name. Use this function when a statement or function needs the name of a location but whose name-index number is stored in an attribute, variable, or some other expression. It can also be used to vary a selected location based on the name-index number stored in the expression. When used in a string expression, as in the second syntax example above, ProModel will convert the name-index number to the actual name of the location. If the expression points to a unit of a multi-unit location, then the LOC() function will return the name of the parent location.

Valid In

Any statement where a location name is normally used, including the Destination field of the Routing edit table. Also used in string expressions.

Components

<location name-index number>

The name-index number of the desired location. This component may be an expression, allowing the location to vary as the simulation progresses. Real numbers will be truncated to integers.

Example

Suppose there are five locations which appear one after the other in the Location edit table as follows: Dock1, Dock2, Dock3, Dock4, Dock5. Shipments could be ordered to each of the five locations in rotation with the following logic. Note that Dist() is a user-defined distribution that returns an integer value for the number of Shipments to order.

```
INT Var1 = 1
WHILE Var1 <= 5 DO
BEGIN
ORDER Dist() Shipments TO LOC(Var1)
INC Var1
END
```

See Also

ENT(), RES(), and LOCATION().

LocState()

General System Function

Syntax samples

LOCSTATE (<locationname>)

Description

Returns a value indicating the current state of the specified location. Return values will range from 1-7, which translate to the following:

1 = idle/empty
2 = setup
3 = operating
4 = blocked
5 = waiting
6 = up (multi-capacity location)
7 = down

Valid In

Any Logic.

Components

<locationname>

The name of the location.

Location()

Location-Specific System Function

Syntax samples

LOCATION()

Attr1 = LOCATION()

IF LOCATION() = 2 THEN WAIT 4 MIN

Description

Returns the current location's name-index number. This function is especially useful in macros and subroutines that vary depending on which location's logic calls them. By using a LOCATION() function with an IF...THEN statement, the macro or subroutine can act differently depending on the location that called it. Also, the same technique can be used to determine which location is carrying out a process when ALL is used as the process location.

Valid In

Any logic.

Example

The individual ovens at a pizza parlor have different capacities but are otherwise exactly the same. While each oven could be modeled as an individual location, it would be easier to model this situation with a multi-unit location. This example uses CAP() and LOCATION() in the processing logic of the parent location to accumulate enough Orders to fill each oven.The individual ovens of a ceramics plant have different capacities, but are otherwise exactly the same. While each oven could be modeled as an individual location, it would be easier to model this situation as a multi-unit location. This example uses CAP() and LOCATION() in the processing logic of the parent location to accumulate enough Sinks to fill each oven.

Process Table

	Location	Operation (min)
Sink	Oven	ACCUM CAP(LOCATION())

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Sink	Cool	FIRST 1	MOVE FOR 5

See Also

LOC() and ENTITY().

Log

General Action Statement

Syntax samples

LOG <string>, <expression>

LOG "Activity Time", Attr1

Description

ProModel assumes that the time stored in the expression is in the model's default time units set in the General Information dialog box. Use the LOG statement to record the time from one statement to another statement by storing the time of the first statement in an attribute, variable, or array element with CLOCK() and using the same attribute, variable, or array element as the expression in the LOG statement. Use the LOG statement to determine throughput time or throughput in a particular section of the facility.

Valid In

Downtime logic, location processing logic, node entry and exit logic, and routing exit logic.

Components

<string>

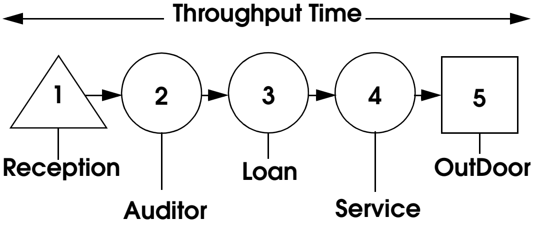
This string will be stored in the file before the result of the log expression and may not be a string expression. Use this string to identify the number that follows.

<expression>

The result of this expression subtracted from the model's current time will be stored in the file <model name>.LAP, after the string above.

Example

The example below shows a LOG statement used to capture each Client's total throughput time as they travel through the system. Time starts when the Client arrives at Receptionist and stops when the Client exits through the door. The first process sets attribute CT equal to the current clock time. Next, Clients are sent randomly to one of three offices. Finally, when a Client leaves the system at location OutDoor, the LOG statement records the cycle time by subtracting the time stored in attribute CT from the current simulation time.



Process Table

	Location	Operation (min)
Client	Reception	CT=CLOCK()
Client	Auditor	WAIT T(5,6,8)
Client	Loan	WAIT T(4.5,5,7)
Client	Service	WAIT N(6.2,1.1)
Client	OutDoor	LOG “Cycle Time”, CT

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Client	Auditor Loan Service	RANDOM 1 RANDOM RANDOM	MOVE FOR 1.5
1	Client	OutDoor	FIRST 1	MOVE FOR 0.5
1	Client	OutDoor	FIRST 1	MOVE FOR 0.5
1	Client	OutDoor	FIRST 1	MOVE FOR 1.2
1	Client	EXIT		

See Also

WRITE, WRITELINE, and READ. Also see “External Files” on page 262.

MapArr

General Action Statement

Syntax samples

```
MAPARR <array name>{TO <variable name>}
MAPARR Array1 TO Var10
MAPARR Array5
```

Description

Starting with the variable you specify, the MAPARR statement maps each individual cell of an array to a unique variable (i.e., if you define 12 cells for the array, the array will map to 12 variables). To display the cell value of a mapped array, create a variable graphic for the variable to which you mapped the array cell. ProModel collects statistics for an array cell through the variable to which you mapped the cell. (Choose "Basic" or "Time Series" statistics for a mapped variable, then view the variable in the Statistics Output program.)

If you do not specify the optional variable name in the statement, ProModel will unmap the array from the variables to which you originally mapped it. You can remap arrays by using the MAPARR statement again.

Valid In

Any logic.

Components

<array name>

Map, unmap, or remap this array. The brackets, [ ], are unnecessary after the array name in the statement.

TO <variable name>

The optional name of the variable to which you map the first cell in the array. If you do not specify a name, ProModel will unmap the array from the variables.

Example

Suppose you want to dynamically view an array, Storage_Array, during simulation. The array has a dimension of 2x3x2 (a three-dimensional array with 2 cells in the first dimension, 3 cells in the second, and 2 cells in the third) and contains a total of 12 cells (multiply all the dimensions together).

Since you already used the first 8 of the 30 variables defined in the model, Var1 through Var30, you will start mapping the array with Var9 and end with Var20 (the 12th variable from Var9 listed in the Variables module). In the initialization logic, use the following statement:

```
MAPARR Storage_Array TO Var9
```

The cells in Storage_Array will map to variables Var9 to Var20 in the following order:

```
[1,1,1] ... Var9
[1,1,2] ... Var10
[1,2,1] ... Var11
[1,2,2] ... Var12
[1,3,1] ... Var13
[1,3,2] ... Var14
[2,1,1] ... Var15
[2,1,2] ... Var16
[2,2,1] ... Var17
[2,2,2] ... Var18
[2,3,1] ... Var19
[2,3,2] ... Var20
```

In the Variables module, create graphics for variables Var9 through Var20 and place them on the layout. This will allow you to view them during the simulation.

Please note

Changing the cell value of a mapped array will change the value stored in the array cell AND the value of the variable to which you mapped the specific cell. Changing the value of a variable (e.g., INC Var12), however, will change ONLY the variable value and NOT the cell value of the mapped array cell. Also, when you use a mapped array in an expression, the array returns the value of the variable mapped to it.

Match

Entity-Related Operation Statement

Syntax samples

MATCH <attribute>

MATCH Attr1

Description

Causes the current entity to wait until the value of the specified attribute matches the value of another entity's same attribute. Both entities must have a corresponding MATCH statement specified for the same attribute name. The two entities may be at any location in the model, including the same location and two completely unrelated locations. Therefore, the value of the attribute to MATCH should almost always be assigned to the value of a global variable incremented for each entity that will be matched, as in the following example.

Locations using the MATCH statement usually should be multi-capacity because otherwise they would not be able to process any other entities until the MATCH was made. Additionally, locations using MATCH usually should be non-queuing to allow the entities to move out of sequence when the MATCH is fulfilled.

Use the MATCH statement to pair two specific parts before assembling them, or to match a work order to a completed job.

Valid In

The operation column of process edit tables only.

Components

<attribute>

Any attribute associated with the processing entity.

Example

You can use a MATCH to recombine the parts of an entity which has been split using multiple routing blocks. In the example below, every time an EntAB arrives at the location, the variable Count increments. ProModel assigns Attr1 for EntAB the value of count, which ensures that each EntAB will have a unique value for Attr1. Additionally, when each EntAB is split into EntA and EntB, both of the resulting entities, (EntA and EntB) will have the same value for Attr1. EntA and EntB then travel independently through their respective processing steps, denoted by the ellipses. Finally, EntA and EntB arrive at Loc10A and Loc10B respectively, where each piece must wait for its matching half to arrive before consolidating the entities with the JOIN construct.

Process Table

	Location	Operation (min)
EntAB	Loc1	INC Count Attr1= Count
EntA	Loc2	...
EntA	Loc10A	MATCH Attr1
EntB	Loc3	...
EntB	Loc10B	MATCH Attr1 JOIN 1 EntA

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	FIRST 1	MOVE FOR 5
2	EntB	Loc3	FIRST 1	MOVE FOR 5
...	...	...	...	...
1	EntA	Loc10B	JOIN 1	MOVE FOR 30 sec
...	...	...	...	...
1	EntAB	Loc11	FIRST 1	MOVE FOR 5

See Also

WAIT...UNTIL and LOAD.

Move

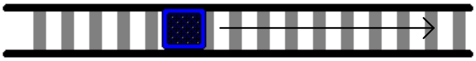
Entity-Related Operation Statement

Syntax samples

```
MOVE {FOR <time expression>}
MOVE FOR .25 min
MOVE FOR 3.2
MOVE
```

Description

Moves the entity to the end of a queue or conveyor location. Use the MOVE statement to explicitly control the movement of an entity through a queue or conveyor.



Please note

If there is no MOVE statement, when an entity enters a queue or conveyor, it executes all the processing logic defined for the entity at that location and then moves to the end of the queue or conveyor. For queues, its movement time is based on the entity's speed and the queue's length. For conveyors, its movement time is based on the conveyor speed and length.

If an entity processing at a queue or conveyor encounters a MOVE statement, the entity stops executing the processing logic, moves to the end of the queue or conveyor in the appropriate amount of time, and then resumes the processing logic. The move-time for an entity on a conveyor is calculated using the following formula:

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

For queues only, a MOVE may optionally be followed by a FOR statement and then a move-time. If a move-time is specified, the entity moves through the queue in the specified amount of time regardless of the entity's speed and the queue's length. Entities with shorter move times will catch up to, but will not pass, entities with longer move times.

If a queue is not empty when an entity enters the queue, then the move-time will continue to elapse even though the arriving entity's graphic may have stopped moving. When an entity's move-time has elapsed, an entity will begin executing any logic following the MOVE statement and then will be available for routing, even if graphically the entity does not appear to be at the end of the queue.

For a conveyor, if additional logic follows a MOVE statement, the entity must advance to the last position on the conveyor before the remaining logic is executed.

Valid In

The operation column of process edit tables only, and only if the location is a queue or a conveyor. MOVE may only be used once per logic.

Components

<time expression>

The amount of time needed for the entity to travel through the queue. This expression is ignored for conveyors. It is evaluated every time the statement is encountered and should be on the same line as the MOVE command.

Example

The example below shows the processing steps necessary to mimic the behavior of a queue system using MOVE statements. Locations Queue1, Queue2 and Queue3 should each be represented graphically with a queue symbol. The time value in the MOVE statement represents the time required to traverse each queue section from beginning to end when the queue is empty.

Process Table

	Location	Operation (min)
ALL	Queue1	MOVE FOR 2.5
ALL	Queue2	MOVE FOR 3.5
ALL	Queue3	...

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	ALL	Queue2	FIRST 1	
1	ALL	Queue3	FIRST 1	
...	...	...	...	...

See Also

“Locations” on page 96. Also see WAIT.

Move For

Entity-Related Move Logic Statement

Syntax samples

MOVE FOR <time>

MOVE FOR 0

MOVE FOR 2.5 + CleanupTime

MOVE FOR N(8, .5) + 3 sec

Description

Used to specify the amount of time required to move the entity. A move-time of zero may be entered to cause events for other entities occurring at the same simulation time to be processed before any additional logic is processed for the current entity. If no move related statement (MOVE FOR, MOVE ON, MOVE WITH) is specified, the entity instantly enters the next location and immediately begins executing the operation logic for that location.

Valid In

This statement is valid in Move Logic. MOVE FOR may be encountered only once by an entity in the same logic.

Components

<time>

The length of time the system takes to execute the move. This expression is evaluated whenever the statement is encountered. If no time unit is specified, the default time unit specified in the General Information dialog is applied.

Example

It takes 4 minutes for the entity, Cutter, to move from one location, Grinder, to the next location, Profiler.

Process Table

	Location	Operation (min)
Cutter	Grinder	GET Operator WAIT 1
Cutter	Profiler	WAIT Attr1

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Cutter	Profiler	FIRST 1	MOVE FOR 4
1	Cutter	Exit		

See Also

MOVE ON and MOVE WITH. Also see “Routing Move Logic” on page 302.

Move On

Entity-Related Move Logic Statement

Syntax samples

MOVE ON <path network>

MOVE ON StatPath2

Description

Use this statement to move an entity along a path network.

Valid In

This statement is valid only in Move Logic. MOVE ON may only be encountered once by an entity in the same move logic.

Components

<path network>

Any valid path network name.

Example

An entity, EntA, moves from StationA to StationB along a network called Net1.

Process Table

	Location	Operation (min)
EntA	StationA	WAIT 2
EntA	StationB	...

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	StationB	FIRST 1	MOVE ON Net1
...	...	...	...	...

See Also

MOVE FOR and MOVE WITH. Also see “Routing Move Logic” on page 302.

Move With

Entity-Related Move Logic Statement

Syntax samples

```
MOVE WITH <res1> {,p1}
   OR <res2> {,p1}
   {FOR <time >} {THEN FREE}
MOVE WITH Technician, 100
MOVE WITH Operator1, 399 FOR 3 min
MOVE WITH Truck1, 99 THEN FREE
MOVE WITH Operator1 OR Operator2
```

Description

This statement is used to move an entity using a designated resource such as a person or forklift. With the OR operator, you can designate alternative resources for making the move. In this case, the statement captures the first available resource designated in its expression and makes the move. As soon as the destination becomes available, the entity implicitly gets the resource. However, if one of the resources is already owned by the entity, it will use that resource.

It also allows you to set the priority (p1) for accessing the designated resource. If the resource is already owned by the entity, this priority is ignored.

If the resource is static, you may specify a time (FOR <time expression>) for the move. If a resource is dynamic, a time (FOR <time expression>) is not valid. If you use “FOR <time>” with a dynamic resource, ProModel ignores the time. The resource will travel based on either the time or speed/distance defined in the path networks module.

The resource used to make the move is freed only if the THEN FREE option is used.

Valid In

This statement is valid only in Move Logic. MOVE WITH may only be encountered once by an entity in the same move logic.

Components

<res1>

Resource to be captured and used to transport the entity.

<res2>

Alternate resource to be captured and used to transport the entity.

<priority>

The priority for accessing the resource. If the resource is already owned by the entity, this priority is ignored.

<time>

The length of time the system takes to execute the move. Used only if the resource is static. This expression is evaluated whenever the statement is encountered. If no time unit is specified, the default time unit specified in the General Information dialog is applied.

Example

An entity moving from Station A to Station B may use either Tech1 or Tech2 to move the entity depending on which one is available first. The resource is freed after the move.

```
MOVE WITH Tech1 OR Tech2 THEN FREE
```

The same thing could be accomplished in longer form:

```
GET Tech1 OR Tech2
MOVE WITH OwnedResource()
FREE OwnedResource()
```

See Also

MOVE FOR and MOVE ON. Also see “Routing Move Logic” on page 302.

Next()

Resource-Specific System Function

Syntax samples

NEXT()

Var1=NEXT()

IF NEXT() = PathNet5.N11 THEN Var5=3

Description

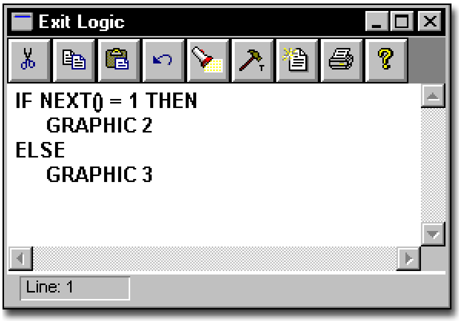
Returns the name-index number of the resource's destination node. Use NEXT() to determine the direction an entity is headed and choose the appropriate graphic. This function can be used to control interference between multiple transporters on the same path network. You can also check the name-index number of the next node by specifying <path network name>. <name of the node>. For example, if you wanted to know if the next node is N5 on the network Net3, you could specify "IF NEXT() = Net3.N5 THEN..." in the node exit logic.

Valid In

Node exit logic. This function returns a name-index number.

Example

This Exit Logic window shows that whenever the resource leaves a node a check is made to see if the name-index number of the next node equals 1. If so, the resource graphic is changed to Graphic 3. (Otherwise it is changed to Graphic 2.)



```
IF NEXT() = 1 THEN
   GRAPHIC 2
ELSE
   GRAPHIC 3
```

See Also

LAST() and WAIT...UNTIL.

Order

General Action Statement

Syntax samples

ORDER <expression> <entity> {TO <location>}

ORDER 10 EntA TO Loc2

ORDER Order_Qty_Attr ENT(Entity_Attr) TO LOC(Loc_Attr)

Description

Causes the specified number of entities to be created and placed into the system at the designated location. If the location does not have enough capacity for all the new entities, the excess entities will be destroyed. Attributes from the creating entity will be copied to the resulting entities. Use ORDER to replenish inventories when a particular condition occurs. Such as when an inventory reaches the minimum level.

Valid In

Any logic.

Components

<expression>

The number of new entities to be ordered. This field is evaluated every time the ORDER statement is encountered, allowing the number ordered to vary as the simulation progresses.

<entity>

The name of the new entities. ENT() may be used for an entity name.

<location>

The destination of the new entities. LOC() may be substituted for the names of locations. If no location is specified, the entities will be ordered to the location of the ORDER statement.

Example

In this example, EntA arrives at LocA1 and triggers an order for 100 EntB's to LocB1.

Process Table

	Location	Operation (min)
EntA	LocA1	ORDER 100 EntB TO LocB1
EntB	LocB1	...

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	LocA	FIRST 1	MOVE WITH Forklift

See Also

SEND, CREATE, and SPLIT AS.

OwnedResource()

Resource-Specific System Function

Syntax samples

```
OWNEDRESOURCE ({<expression >})
OWNEDRESOURCE (2)
OWNEDRESOURCE (ResQty())
OWNEDRESOURCE ()
```

Description

Returns the name-index number of the nth resource currently owned by the entity or downtime making the function call. The function parameter indicates the position of the resource in the chronological list of owned resources. For example, OwnedResource(1) returns the longest owned resource in the list and so on.

When used without a parameter, this function returns the most recent resource that the entity captured and still owns. If the parameter value is not within the range of the resource list, or if the entity or downtime currently does not own a resource, the function will return a 0 (zero) without a warning or error message.

A preempted resource is NOT removed from the list but marked temporarily to indicate that the preemptee does not own the resource. After the resource resumes the original process after being preempted, it retains its original rank in the list.

Valid In

Entity speed fields, traveling-time fields, resource fields, resource downtime logic, location processing logic, location downtime logic, routing fields, arrival logic, debug user-condition fields, and move logic.

Components

<expression>

The nth resource currently owned by the entity or downtime making the function call. When this parameter is not used, the function returns the last resource captured that is still owned by the entity.

Example

Suppose an entity owns two resources. The first resource was captured using the statement GET Worker. The second resource was captured using the statement GET Oper1 OR Oper2. We know the entity owns the resource Worker, but we do not know if the entity owns Oper1 or Oper2. It is time to release the second resource so we use the following logic:

```
FREE OWNEDRESOURCE()
```

See Also

RES(), GET, JOINTLY GET, USE, and FREE. Also see “Resources” on page 132.

Pause

General Action Statement

Syntax samples

PAUSE {<string expression>}

PAUSE

PAUSE “Var1 =” $ Var1

PAUSE “Reached the midpoint of the simulation.”

Description

Pauses the simulation and optionally displays a message at some point of interest. This pause allows the user to examine the system in detail. An information dialog box will appear on the animation screen when the pause occurs. The simulation will continue only when the user selects Resume Simulation from the Simulation menu.

Valid In

Any logic.

Components

<string expression>

The optional message to display.

Example

The simple example below pauses the simulation after the 100th EntA has been processed at Loc1. The purpose for doing this might be to view the current state of the system at this particular point in time.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT N(3.2,.1) INC Total IF Total >= 100 THEN PAUSE

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	FIRST 1	

See Also

STOP, DISPLAY, and PROMPT.

PercentOp()

General System Function

Syntax samples

PERCENTOP (<locationname>)

Description

Returns the cumulative operation time percentage for the specified, single-capacity location. The value returned by this function represents the cumulative percentage of time the location was actually processing an entity up to the point where the function was called.

If PercentOp() is called for a multiple-capacity location, the value returned will always be zero, since operation time percentage is not calculated for multiple-capacity location

Note: The method used to calculate operation percentage for this function is the same method used in the output statistics.

Valid In

Any Logic.

Components

<locationname>

The name of the location.

PercentUtil()

General System Function

Syntax samples

PERCENTUTIL (<locationname>)

Description

Returns the cumulative utilization percentage for the specified location. The value returned by this function represents the cumulative percentage of capacity occupied at the location, on average, at the time the function was called.

$$\frac{\text{Cumulative Occupancy Time } \times \text{ } 100}{\text{Capacity } \times \text{ Scheduled Time}}$$

Cumulative Occupancy Time in the above equation refers to the sum of the clock time each entity spends at a location for processing.

PercentUtil() may be called to return percent utilized for both multi- and single-capacity locations.

Note: The method used to calculate utilization percentage for this function is the same method used in the output statistics.

Valid In

Any Logic.

Components

<locationname>

The name of the location.

Preemptor()

Preemption Logic System Function

Syntax samples

```
PREEMPTOR()
Var1=PREEMPTOR()
```

Description

Identifies whether a downtime or entity is making the preemptive request. The function returns the name index number of the preempting entity; however, it returns a 0 if the preemptor is a downtime.

Valid In

Operation logic defined as a preemptive process.

Example

Workers may process an entity, bracket, at one of two locations, Punch_1 or Punch_2. You may preempt the bracket at Punch_1 by either an entity with a preemptive priority or a location shift downtime with a preemptive priority. If an entity preempts the bracket, workers send the bracket to Punch_2 to finish processing. Punch_1 and Punch_2 are on the same shift, and are scheduled to go off-shift at the same time. Therefore, if a shift downtime preempts the bracket at Punch_1, the bracket would not be able to go to Punch_2. In this case, the bracket is sent to a location called Punch_Wait where it waits for Punch_1 to come back on-shift. The PREEMPTOR() function determines whether an entity or shift downtime preempts the bracket.

Process Table

	Location	Operation (min)
Bracket	Punch_1	WAIT 3 min

Bracket*	Punch_1	IF PREEMPTOR()>0 THEN
		  ROUTE 1
		ELSE
		  ROUTE 2

* Preemptive Process Record

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Bracket	Inspect	FIRST 1	
1	Bracket	Punch_2	FIRST 1	
2	Bracket	Punch_Wait	FIRST 1	

See Also

“Preemption Process Logic” on page 300.

Priority

Shift & Break Logic Statement

Syntax samples

```
PRIORITY <expression>
PRIORITY 199
```

Description

This statement is used to change the priority of the off-line state of the location or resource. If the priority is less than the value set previously, the system will check if the location or resource can be preempted.

Valid In

Shift and break main logic only. This statement is not valid in pre-off shift or pre-break logic.

Components

<expression>

Any expression that yields a value between 0 and 999. Standard ProModel priority levels apply.

Example

Suppose you want to insure that the resource is non-preemptable for the first four hours it is off-shift. Simply enter a high (e.g., 999) off-shift priority value in the priority dialog in the Shift Assignments module. Enter the following off-shift logic to lower the priority (to 99) four hours into the off-shift period:

```
WAIT 4 hr
PRIORITY 99
WAIT 4 hr
SKIP
```

See Also

“Shift & Break Logic” on page 305.

Prompt

General Operation Statement

Syntax samples

```
PROMPT <string expression>, <name>{,
    <choice1>:<expression1>,
    <choice2>:<expression2>,
    <choice3:<expression3>...}
PROMPT "Enter the number of entities to process:", Var2
PROMPT "Enter the size of batches to accumulate:",Var1, "Large": 20, "Medium":
    15, "Small": 10
```

Description

Pauses the simulation and displays either a message and input field or a menu for selecting a choice. The value entered or selected is then assigned to the designated variable, array element, or attribute. To have PROMPT present a menu, specify one or more choices as in the second syntax example above. The value already in the variable, array element, or attribute is used as the default value in the dialog box. One use of PROMPT is to give the user the option to change the operation time represented by a variable during a simulation.

Valid In

Any logic.

Components

<string expression>

The message to display. This expression should tell the user what value to enter or choose.

<name>

The name of the variable, array element, or attribute to give the value. The value already in this item will be used as the default value.

<choices>

A string expression identifying the choice. Any number of choices may be specified, and all must have corresponding expressions.

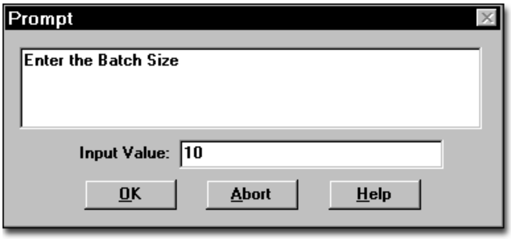
<expressions>

The value to assign the variable, array element or attribute if the user selects the corresponding choice from the menu.

Example

The logic below uses PROMPT to let the user select any size of a batch. Attr1 represents the batch size for EntA. If the batch size has not been set (if Attr1=0), then the user is prompted to enter the batch size. The SPLIT AS statement then splits the single entity into the number of entities specified in the prompt dialog box. The PROMPT statement in this example displays the following dialog box.

```
IF Attr1=0 THEN
PROMPT "Enter the Batch Size", Attr1
SPLIT Attr1 AS EntA
```



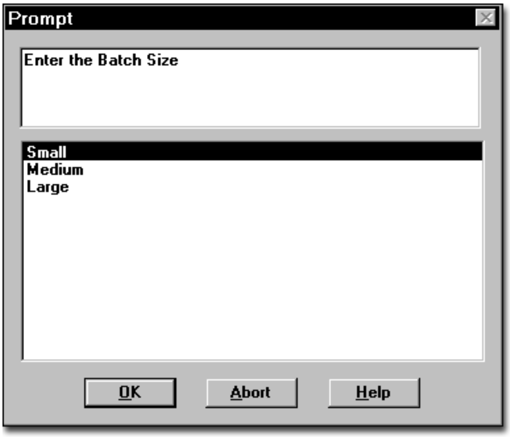
Enter the Batch Size

Input Value: 10

Example

This logic works similarly to the logic in example one, except that it uses PROMPT to let the user select from one of three different sized batches. The PROMPT statement in this example displays the dialog box below.

```
IF Attr1=0 THEN
PROMPT "Enter the Batch Size", Attr1, "Small":10,
"Medium":15, "Large":20
SPLIT Attr1 AS EntA
```



Enter the Batch Size

See Also

PAUSE; DISPLAY. Also see "Run-Time Interface" on page 242.

Rand()

Math Function

Syntax samples

RAND(<expression>)

RAND(10) min

ORDER RAND(10) EntA TO Loc1

IF RAND(100) > 45 THEN ROUTE 1

Description

Returns a random value n between 0 and X (0 <= n < X) where X is the result of the expression.

To generate random numbers between the expression and a number other than zero use the expression as the range between the maximum and minimum values. Then add the minimum to the random number. For example, the statement Attr1 = 2+RAND(3) generates a random number from two up to, but not including, five. An alternate method is to use a uniform distribution. Note that the RAND() function works very similar to a uniform distribution. For more information on Distributions, see “Distribution Functions” on page 437.

This function returns a real number, although the real value may be converted to an integer. To have the RAND() function generate random integer values between zero and some upper bound, use the formula, Integer_Value = RAND(X+1), where X is the greatest integer that should be returned by the function. For example, to generate a random integer value between zero and six, use the formula, Integer_Value = RAND(7). The RAND(7) part of this formula will actually generate a real value between 0 and 6.999, but will be truncated to a whole number between 0 and 6 when assigned to an integer value. Therefore, when generating integer values, make sure that the result of the RAND() function is assigned to an integer value, or is used in an expression where it will be truncated to an integer.

Valid In

Any expression.

Components

<expression>

The upper limit of the random value to return. This value will never be returned.

Example

Two resources, Oper_1 and Oper_2, perform the same task at a location, Press. Oper_1 works at several other locations. As a result, Press only uses him 30% of the time. The other 70% of the time, Oper_2 performs the task.

Process Table

	Location	Operation (min)

Axle	Press	IF RAND(100) < 30 THEN
		USE Oper_1
		ELSE USE Oper_2

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Axle	Lather	FIRST 1	MOVE FOR 3

See Also

“Streams” on page 266.

Read

General Action Statement

Syntax samples

```
READ <file ID>, <name>
READ File1, Var1
```

Description

Reads the next numeric value from a general read file and assigns that value to a name. General read files are defined in the External Files Editor. When reading from a file, ProModel skips all non-numeric data, such as text, and reads the next numeric value. Thus comments and notes may be included in a read file. Multiple replications of a model will continue reading from a file where the previous replication left off unless reset with the RESET statement.

Please note

ProModel reads a period (.) in a General read file as a zero. To avoid this, you should use the comment symbol (#) in front of notes and comments that contain a period.

READ can read ASCII files. Most spreadsheet programs can convert spreadsheets to ASCII files (.TXT) and comma-delimited files (.CSV).

If a read statement is not assigning the right values to the proper name, there may be numeric information in the read-file's header information (notes and comments). Additionally, if the values are being read into an array, the array indices may not be incremented properly between reads.

If a read file should be read more than once in a model, it may need to be reset. One way to tell when a file needs to be reset is with an end of file marker such as 9999 and the following two lines of logic.

```
Read File1,Value
IF Value= 9999 then RESET File1
```

Valid In

Any logic.

Components

<file ID>

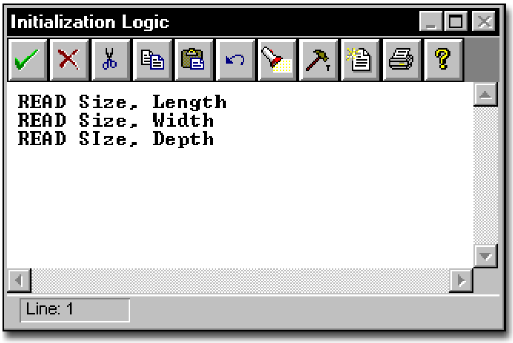
The file identifier as defined in the External Files Editor.

<name>

The variable, array element, or attribute to be assigned the value.

Example

The example below shows an outside file being read in the Arrival logic for entity type Box. In this case three values representing the length, width and depth of each Box are listed on each line of a file called "Size.Dat." The File ID for this file is simply Size. Length, Width, and Depth are all entity attributes.



```
READ Size, Length
READ Size, Width
READ SIze, Depth
```

See Also

“External Files” on page 262. Also see RESET and CLOSE.

Real

Local Variable Declaration Statement

Syntax samples

```
REAL <name1>{= <expression1>, <name2>= <expression2>...)

REAL Var1

REAL Counter = 0

REAL Var1 = CLOCK(SEC), Random_Num = RAND(10)
```

Description

Creates a local variable of type real. Local variables work much the same as attributes, except that they only are available within the logic that declares them. A local variable will be created for each entity that encounters a REAL statement. Local variables are not directly available to subroutines, which have their own local variables. However, a local variable may be passed to a subroutine as a parameter. Local variables are available to macros.

Valid In

Any logic. Variables declared with REAL are valid in any expression within the logic where a real number is valid.

Components

<names>

An identifier for the first local variable. This identifier must be a valid name.

<expressions>

The variable will initially be assigned this value. This expression is evaluated every time the REAL statement is encountered.

Example

The example below uses a local real variable to track the total time an entity waits for another entity to be joined to it. A shipping area has an operation where invoices are joined to boxes to produce packages. The user wants to know exactly how long an invoice must wait before a box arrives to be joined. By setting the value of a local variable, Start, to the clock time just before the JOIN statement and using a LOG statement immediately after the JOIN, we can determine how long each invoice had to wait before a box arrived. A local variable is a better choice here than an entity attribute because the only place the information is needed is inside this logic.

Process Table

	Location	Operation (min)
Box	Packing	WAIT N(10,3)
Invoice	Shipping	REAL Start = CLOCK() JOIN 1 Box LOG "Delay:", Start

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Box	Shipping	JOIN 1	MOVE FOR 1
1	Package	Dock	FIRST 1	MOVE FOR 3

See Also

INT. See “Variables” on page 231.

Real()

Type Conversion Function

Syntax sample

```
REAL(<expression>)
Var2 = Var1 + REAL(Var3)
Attr3 = 1.05 * REAL(Var5)
```

Description

Converts an integer to a real number. ProModel automatically converts integers to real when needed.

Valid In

Any expression.

Components

<expression>

REAL() converts this expression to a real number.

See Also

ROUND() and TRUNC().

Rename

Entity-Related Operation Statement

Syntax samples

RENAME {AS} <new entity name>

RENAME EntB

RENAME AS EntB

RENAME AS ENT(Var2)

Description

Renames the processing entity to the new entity name. After a RENAME statement is encountered, the entity then searches forward in the process list and again from the beginning until a process is found at the current location that has been defined for the new entity type. No further logic will be executed for the entity under its original name. Use RENAME to start collecting statistics for an entity under a new name. Usually, the easiest and most efficient way to rename an entity is simply by using the new name as the output entity in the routing.

Valid In

The operation column of process edit tables only.

Components

<new entity name>

The new name of the processing entity. ENT() may be substituted for the entity name.

Explicit Entity Actions

With RENAME, statistics and cost continue on with the entity.

Implicit Entity Actions

ProModel allows you to define the RENAME action *implicitly* as part of the routing definition. To do this, define a route block with a different input and output name and the New Entity option *un*checked.

Example

The following example shows how two entities, EntA and EntB, join together at Loc2. Once the join is complete, ProModel renames the resulting entity EntC and processes it according to a normal distribution N(9.4,.3). RENAME is the last statement in the process because as soon as you rename an entity, ProModel searches the processing logic for a process for the entity with the new name.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT 1.5 min
EntB	Loc2	JOIN 1 EntA RENAME AS EntC
EntC	Loc2	WAIT N(9.4,.3)

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	JOIN 1	
1	EntC	Loc3	FIRST 1	

See Also

COMBINE, SPLIT AS, and GROUP.

Report

General Action Statement

Syntax samples

REPORT {WITH RESET} {AS <string expression>}

REPORT

REPORT WITH RESET

IF thruput = 50 THEN REPORT AS "RepOvr50"

Description

Calculates and reports the current statistics to the output database. This is useful to get a snapshot of the model while it is running.

The REPORT statement may be followed by the WITH RESET option to reset the statistics after the report is made to the database When you use the WITH RESET option, you generally want to provide some looping or event creation that will call the report function at the appropriate time.

Used with the AS option, REPORT creates a report with the name specified in the expression that can be accessed in the Output Program when creating a General Stats report.

Valid In

Any logic.

Components

<string expression>

A unique name given to the report so it can be easily identified in the General Stats dialog in the Output Program. If any reports have the same name, a number is tacked on the end of the name to make it unique.

Example

To get a snapshot report every 40 hours, enter the following:

```
WHILE Clock(hr) < 10000 DO
BEGIN
WAIT 40 hr
REPORT AS "40HOUR"
END
```

This results in reports named, 40HOUR, 40HOUR2, 40HOUR3....

Please note

If you use the REPORT statement even once, a final overall report will NOT be created since the report generated with your use of the statement may be the final report desired. In this case, you must use the REPORT statement at the end of termination logic in order to create a final report if desired. If a REPORT statement is never used, a final overall report is created automatically.

See Also

RESET STATS and WARMUP.

Res()

Name-Index-Number Conversion Function

Syntax samples

```
RES(<resource name-index number>)
USE 10 RES(Var1) for 1.5 min
FREE RES(Var1)
DISPLAY "Now using" $ RES(Var1)
```

Description

Converts a name-index number or integer to a resource name. Use this function when a statement or function needs the name of a resource whose name index number is stored in an attribute, variable, or some other expression. When used in a string expression expecting, such as in the third syntax example above, ProModel will output the actual name of the resource.

Use RES() to assign a properly skilled operator according to the attribute of the part or to change the duties of resources as the simulation progresses.

Valid In

Any statement where a resource name is normally used, except in the Move Logic field in the Routing edit table. Also used in string expressions.

Components

<resource name-index number>

The name-index number of the resource desired. This component may be an expression. Real numbers will be truncated to integers.

Example

The logic below uses five different resource types for ten minutes in rotation.

```
INT Var1 = 1
WHILE Var1 <= 5 DO
BEGIN
USE RES(Var1) FOR 10 min
INC Var1
END
```

See Also

ENT(), LOC(), and GRAPHIC.

Reset

General Action Statement

Syntax samples

RESET <file ID>

RESET Times

RESET (Times)

Description

Starts a general read file over from the beginning. RESET is used primarily in the Initialization or Termination logic to reset a general read or write file at the beginning or end of multiple replications and single, independent runs. RESET can also be used to re-read cyclic data in the same simulation run. The parentheses are optional and are included only to insure compatibility with older models.

Valid In

Initialization and termination logic, node entry and exit logic, down-time logic, location processing logic, routing exit, and arrival logic.

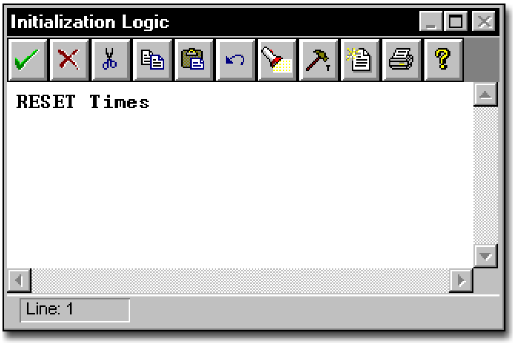
Components

<file ID>

The file identifier as defined in the External Files Editor.

Example

The example below shows how a general read file, Times, is reset in the Initialization logic. Each time a simulation begins, whether a single replication or one of several multiple replications, the Times file is reset so that calls to the Times file start at the beginning of the file.



```
RESET Times
```

See Also

"External Files" on page 262. Also see READ, WRITE, XWRITE, and RESET.

Reset Stats

General Action Statement

Syntax samples

```
RESET STATS
IF Total = 20 THEN RESET STATS
```

Description

Resets the simulation statistics. Useful in connection with the REPORT statement to manually control statistics for reporting purposes in case specific or event logic.

Valid In

Any logic.

Example

Suppose you want to generate a new output report for each 20-hour period of the simulation. Enter the following logic in an independent subroutine which is activated at the beginning of the simulation:

```
WHILE Clock(hr) < 10000 DO
BEGIN
WAIT 20 hr
REPORT
RESET STATS
END
```

See Also

REPORT and WARMUP.

Resource()

Shift & Break System Function

Syntax samples

RESOURCE ()

Description

Returns the name-index number of the resource currently processing the off-shift or break logic.

Valid In

Off-shift and break logic.

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_To_Process = 0
END
END
```

See Also

LOCATION(), FORLOCATION(), and FORRESOURCE().

ResourceUnit()

Shift & Break System Function

Syntax samples

```
RESOURCEUNIT()
```

Description

Returns the unit number of the resource being used.

Valid In

Off-shift and break logic.

Example

When a multi-unit resource goes on shift, it is sometimes helpful to know which unit of the resource is going off shift. Depending on the resource unit you may want to skip the shift or update a variable that is used elsewhere in your model.

ResQty()

Entity-Specific System Function

Syntax samples

RESQTY({<resource name>})

IF RESQTY(Res1) > 5 THEN FREE 5 Res1

Description

Returns the number of units of the specified resource that the current entity owns. RESQTY() can be used to determine the amount of time necessary to process an entity based on the number of resources an entity owns.

Valid In

Entity speed fields, traveling-time fields, resource fields, location processing logic, routing fields, arrival logic, debug user-condition fields, and exit logic. This function returns an integer.

Components

<resource name>

The name of the resource to check for. If no resource name is specified, the total number of units of all resource types owned by the entity is returned. RES() may be substituted for a resource name.

Example

The example below demonstrates the use of RESQTY() to adjust the processing time for an entity based on the number of resources available to process it. Before EntA's arrive at Loc1, they capture a certain number of resources named Res1. Processing logic at Loc1 determines how many Res1's each EntA captured and processes it accordingly. The more resources an entity captures, the more workers available to work on the project, and the less time it takes. ProModel then routes EntA's on to Loc2. The logic at Loc2 makes sure that no EntA owns more than one Res1.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT 120/RESQTY(Res1)
EntA	Loc2	IF RESQTY(Res1)>1 THEN FREE (RESQTY(Res1)-1) Res1

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	FIRST 1	
1	EntA	Loc3	FIRST 1	

See Also

FREECAP(), FREEUNITS(), and GROUPQTY().

Return

General Action Statement

Syntax samples

RETURN {<expression>}

RETURN

RETURN Attr1**Sqrt(Attr2)

Description

Sends a value from a subroutine to the logic that called the subroutine. In the same way that parameters send information from the calling logic to the subroutine, RETURN sends information from the subroutine to the calling logic. After the RETURN is executed, no more logic in the subroutine is executed. When subroutines return values, the RETURN statement must be followed by an expression.

When used in logic that is not a subroutine, RETURN functions like a very powerful BREAK or BREAKBLK statement. Whereas BREAK and BREAKBLK exit only the innermost loop or statement block, RETURN exits the logic completely, no matter how deeply nested inside loops and statement blocks.

Valid In

Any logic, but used most often in user-defined subroutines.

Components

<expression>

The value to return. This expression must be on the same line as the RETURN. If a subroutine was activated, then the return value is ignored.

Example

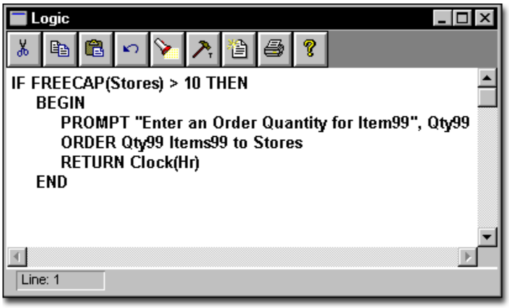
The following example uses a subroutine to check the supply of items in a storage location. If the free capacity of the location is greater than 10, the user is prompted for an order quantity; otherwise no new items will be ordered. If an order is made, the order time is returned; otherwise the variable OrdTm remains unchanged. The logic for Sub1() appears in the logic window.

Process Table

	Location	Operation (min)
Item99	Stores	
Item99	Shipping	WAIT N(3.2,.2) OrdTm = Sub1()

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Item99	Shipping	SEND 1	MOVE FOR 15
1	Item99	EXIT	FIRST 1	



```
IF FREECAP(Stores) > 10 THEN
   BEGIN
      PROMPT "Enter an Order Quantity for Item99", Qty99
      ORDER Qty99 Items99 to Stores
      RETURN Clock(Hr)
   END
```

See Also

“Subroutines” on page 246. Also see BREAK and BREAKBLK.

Round()

Math Function

Syntax samples

ROUND(<expression>)

Integer1 = ROUND(3.5)

Description

Rounds the expression to the nearest whole number. Use this function to override ProModel's default, truncation.

Valid In

Any expression. This function returns an integer.

Components

<expression>

The expression to be rounded.

See Also

TRUNC() and REAL().

Route

Entity-Related Operation Statement

Syntax samples

```
ROUTE <expression>
ROUTE 2
ROUTE Attr1
ROUTE Dist1()
```

Description

Executes a routing block for the processing entity. The process does not continue until all of the entities being routed for the particular block have begun executing their move logic. The processing logic may contain several ROUTE statements. These statements may be selected individually using IF...THEN statements, or they may be processed sequentially, with or without other process statements in between. If any ROUTE statement appears in a process logic, then ProModel assumes all routing for that process will be activated by the user and therefore does no automatic routing. If no ROUTE statement appears in the processing logic, then all routing blocks will be executed automatically once processing logic has been completed.

The ROUTE Statement is most often used with IF...THEN statements to make routing decisions based on complex logic that is not available through other ProModel features (such as system functions or the User Condition routing rule). ROUTE, if used with IF...THEN properly, will insure that only one of the routing blocks is activated.

This statement can be used to route one or more entities and leave behind a “ghost” entity that will process the remaining logic after the route statement. The “ghost” entity is also referred to as the parent entity. The child entity takes the route specified by the ROUTE statement. If the child entity cannot go to the next location and is blocked, the parent entity is also blocked and will not continue logic execution until the child entity is no longer blocked. For more information, see “Entities” on page 118.

Valid In

The operation column of process edit tables only.

Components

<expression>

The integer result of this expression determines the routing block that the entity will take. The expression is evaluated every time it is encountered, allowing the chosen block to vary during the simulation.

Example

This example illustrates a “nested” probability routing. The initial entity, EntAB, enters Loc1 and ProModel makes a decision based on the user-defined distribution Dist1() whether to route the resulting entities according to Route 1, Route 2, or Route 3. If ProModel chooses Route 1, it sends an EntA 60% of the time, or an EntB 40% of the time, to Loc2. If ProModel chooses Route 2, it sends an EntA 20% of the time, or an EntB 80% of the time, to Loc3. If ProModel chooses Route 3, it sends an EntA 30% of the time, or an EntB 70% of the time, to Loc4.

Process Table

	Location	Operation (min)
ENTAB	Loc1	Route Dist1()

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	.600 1	MOVE FOR 2
	EntB	Loc2	.400	MOVE FOR 2
2	EntA	Loc3	.200 1	MOVE FOR 2
	EntB	Loc3	.800	MOVE FOR 2
3	EntA	Loc4	.300 1	MOVE FOR 2
	EntB	Loc4	.700	MOVE FOR 2

See Also

“Routing Rules” on page 415.

Send

General Action Statement

Syntax samples

SEND <expression> <entity name> TO <destination>{,<priority>}

SEND 2 EntA TO Loc2

SEND 1 Grp_A TO Grp_A_Processing, 10

Description

Sends the specified number of a particular entity type to the destination. The entities to be sent must be waiting with a SEND routing rule. The entity that issued the SEND command continues processing whether or not entities of the type requested are waiting to be sent. If no entities are waiting to be sent, a SEND notice is automatically posted so that entities will be sent when they become available.

The SEND statement can model a system based on demand, rather than on entity arrival, (called a pull system). Customer orders cause a SEND to be issued for a main assembly. Main assembly issues SEND commands for sub-assemblies. The example model SEND has an excellent example of this technique.

The SEND statement can also be used as a control device to limit the amount of work-in-progress (WIP) in certain critical areas. Quantities are only sent to the production area when the WIP level falls below a certain point.

Valid In

Any logic.

Components

<expression>

The number of entities to send to the destination. Negative values will generate an error message.

<entity name>

The type of entity to send to the destination. You may substitute ENT() for an entity name.

<destination>

The name of the location to which the entities will be sent. You may substitute LOC() for the location name.

<priority>

Multiple SEND requests for the same entity type are filled according to the longest waiting request having the highest priority. This expression should be a number between 0 and 999. For more information on priorities, see Priorities, at the beginning of this section.

Example

In this example, EntA's arrive at LocA1 and trigger the sending of EntB's to LocB2. The value of EntA's Attr2 determines the number of EntB's sent to LocB2.

Process Table

	Location	Operation (min)
EntA	LocA1	WAIT U(3,.5) SEND Attr2 EntB TO LocB2
EntB	LocB1	Wait 5 min

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	LocA2	FIRST 1	
1	EntB EntB EntB	LocB2 LocB3 LocB4	SEND 1 SEND SEND	

See Also

SEND, ORDER, JOIN, ROUTE, WAIT...UNTIL, and LOAD.

SetRate

General Operation Statement

Syntax samples

SETRATE <resource name>, <expression>, <unit #>

SETRATE Operator, 25, 3

Description

Allows you to define the regular rate of cost for resources contained in a model. If you have already defined the regular rate in the Cost module, this statement will override that rate. You can use SetRate to set different rates for each unit of a resource.

Valid In

Initialization logic.

Components

<resource name>

The name of the resource whose rate you wish to set.

<expression>

The rate assigned to the resource.

<unit #>

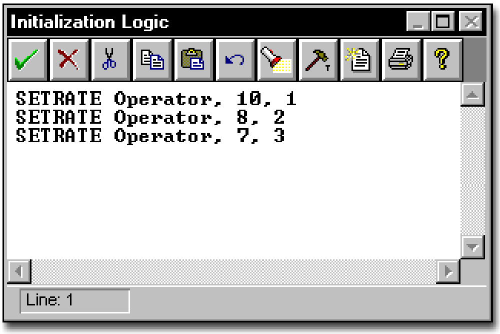
The unit number of the resource. Where multiple instances of a resource exist, you must specify which instance of the resource to use (e.g., Tech1, Tech2, Tech3, etc.). The keyword *ALL* may be used to indicate all instances of a resource.

Please note

SETRATE uses the time units defined for the model. (By default, SETRATE uses hours.)

Example

The logic below displays how you can uniquely assign the regular rate for each unit of a resource, Operator. Operator has three units, meaning that each unit can perform the same task. However, each of the three Operators has a different hourly rate. We set these rates in the Initialization logic using the following:



```
SETRATE Operator, 10, 1
SETRATE Operator, 8, 2
SETRATE Operator, 7, 3
```

See Also

GETCOST, GETRESRATE(), INCENTCOST, INCLOCCOST, and INCRESCOST.

Skip

Shift & Break Logic Statement

Syntax samples

```
SKIP
```

Description

In pre-off-shift or pre-break logic, a SKIP statement causes any off-shift or break main logic to be skipped as well as the off-shift or break time defined in the shift file so that the affected location or resource stays on line.

In off-shift or break logic, a SKIP statement causes the off-shift or break time defined in the shift file to be ignored. This is useful if you want to define your own off-shift or break time as part of the logic rather than use the time defined in the shift file.

Valid In

Shift logic only.

Example

Suppose a Worker is scheduled to go on break at 10:15 for fifteen minutes. However, if there are more than two parts in queue, the Worker will skip his or her break to stay on schedule. The following logic is entered in the pre-break logic for the resource, Worker.

```
IF Contents(Worker_Que) > 2 THEN SKIP
```

See Also

"Shift & Break Logic" on page 305. Also see FORLOCATION() and FORRESOURCE().

Sound

General Action Statement

Syntax samples

SOUND <string expression>

SOUND "Chimes.wav"

SOUND "Tada.wav"

Description

Plays a wavefile. Wavefiles, which have the extension .WAV, may be purchased commercially or created with a sound card. A few sounds, such as the examples here, come with Windows and are found in the Windows directory. Use SOUND to alert a model's user that some event has taken place.

Valid In

Any logic.

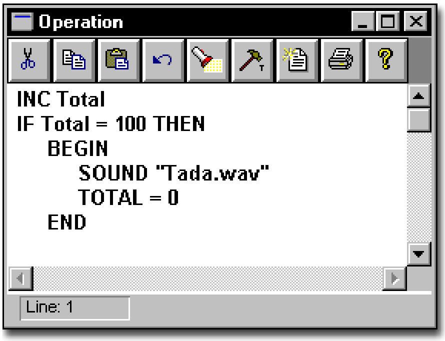
Components

<string expression>

The DOS name of the wavefile to be played. This expression must evaluate to a valid DOS file. It may include a path.

Example

The example below shows an entity's operation logic. A variable called Total is used to keep track of the number of entities passing through the location. Every 100th entity to pass through the location will cause the sound "Tada" to sound, thus notifying the user of the 100th entity. In addition, ProModel resets the counter.



```
INC Total
IF Total = 100 THEN
   BEGIN
      SOUND "Tada.wav"
      TOTAL = 0
   END
```

See Also

PAUSE, DISPLAY, and PROMPT.

Split As

Entity-Related Operation Statement

Syntax samples

```
SPLIT <expression> AS <new entity name>
SPLIT 10 AS Entx
```

Description

Splits an entity into the number of entities you specify, changes the entity names, and divides all cost and time statistics accrued by the base entity between the new entities. ProModel counts the old entity as an exit and the resulting entities share the same attribute values as the original entity.

Any entity you wish to split must release all owned resources using the FREE statement. Use SPLIT AS to divide pieces of raw material into components. The entities formed by the SPLIT AS statement at a location will not appear in the statistics for this location.

Valid In

The operation column of process edit tables only. ProModel does not allow SPLIT AS on conveyors, and not at the end of a queue. You also may not use SPLIT AS after a ROUTE statement. Do not use SPLIT AS in combination with COMBINE, CREATE, GROUP, UNGROUP, LOAD, UNLOAD, or other split statements in the same process logic.

Components

<expression>

Split the entity into this number of entities. ProModel evaluates this expression every time it encounters the statement.

<new entity name>

The name of the resulting entities. Each split entity searches forward in the process list, and then from the beginning of the list, until it finds a process for the new entity type at the current location.

Explicit Entity Actions

When you use the SPLIT AS statement, ProModel divides the accrued cost between the new entities and counts the old entity as an exit. Each new entity begins with new statistical information.

Implicit Entity Actions

ProModel allows you to use the SPLIT AS statement *implicitly* as part of the routing definition. To do this, define a route block with a Quantity field output value greater than 1 and the New Entity option *un*checked.

Example

In the following example, a batch of entities, Batch A, arrives at Loc1 for a 2 hour processing time. Once the processing completes, BatchA splits into individual entities called EntA. ProModel determines the number of EntA's resulting from the SPLIT AS statement by the value of BatchA's attribute, Attr3.

Process Table

	Location	Operation (min)
BatchA	Loc1	WAIT 2 Hr SPLIT Attr3 AS EntA
EntA	Loc1	USE Res1 FOR U(2,.3)

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	FIRST 1	

See Also

JOIN, GROUP, UNGROUP, and CREATE.

Sqrt()

Math Function

Syntax samples

SQRT(<expression>)

Real1 = SQRT(Real2)

Description

Returns the square root of an expression.

Valid In

Any expression. This function returns a real number.

Components

<expression>

SQRT() returns the square root of this expression.

See Also

ROUND().

Please note

To get a root other than the square root, use the exponentiation operator as shown in the following formula:

*X**(1/Y)*

*For example, where Y is the desired root, the formula 9**(1/3) returns the cube root of 9.*

Stop

General Operation Statement

Syntax samples

STOP {<string expression>}

STOP

STOP "Normal termination"

Description

Terminates the current replication and optionally displays a message. The simulation will then continue with the next replication. Use STOP to end a replication when the simulation has run long enough to provide sufficient statistical data.

Valid In

Any logic.

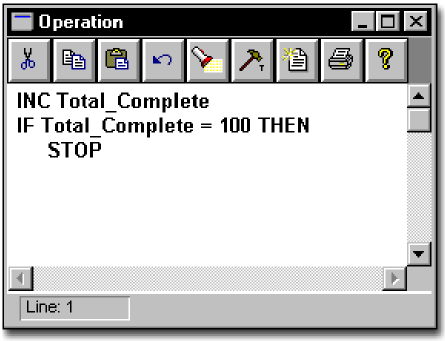
Components

<string expression>

An optional message to display when the replication stops.

Example

In the example below, a STOP statement is used in Operation Logic to end the simulation whenever the variable Total_Complete reaches 100.



```
INC Total_Complete
IF Total_Complete = 100 THEN
   STOP
```

See Also

BREAK, BREAKBLK, RETURN, and PAUSE.

ThreadNum()

General System Function

Syntax samples

THREADNUM()

IF THREADNUM()=215 THEN DEBUG

Description

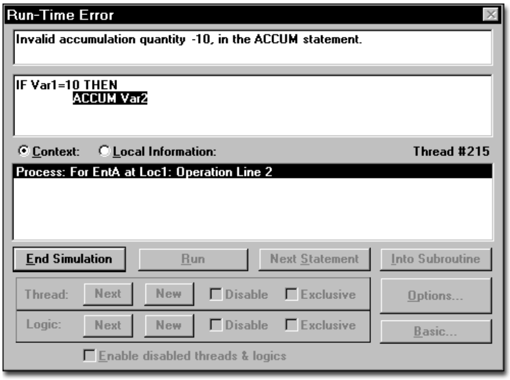
Every time any logic is executed, it is executed by a thread which is assigned a unique number. THREADNUM returns the number of the thread that called the function. This function is most useful in conjunction with the IF...THEN and DEBUG statements to bring up the debugger at a certain process. See below for a detailed example. Note that if the model does not change between simulation runs, every thread will have the same number from run to run, but not from replication to replication. Also, most changes in a model will cause threads to have different numbers on subsequent runs.

Valid In

Any logic

Example

For example, suppose that when the 50th entity of a certain type arrives at a certain location, the model always crashes. A DEBUG statement at the beginning of the processing logic would be cumbersome because the debugger would come up forty-nine times before it was needed. However, when the error occurs, the debugger displays the process and logic that caused the error, as shown below.



Run-Time Error

Invalid accumulation quantity -10, in the ACCUM statement.

```
IF Var1=10 THEN
        ACCUM Var2
```

● Context: ○ Local Information: Thread #215

Process: For EntA at Loc1: Operation Line 2

□ Enable disabled threads & logics

By adding the statement, "IF THREADNUM() = 210 THEN DEBUG" before the statement that causes the error, the simulation will run until the proper process and then bring up the debugger. The debugger can then be used to step through the process to find the particular statement causing the error.

See Also

DEBUG.

TimeLeft()

Preemption Logic System Function

Syntax samples

```
TIMELEFT()
Attr1=TIMELEFT()
```

Description

Returns the time remaining if the preemption occurred during a WAIT statement. The value returned is in default time units and must be checked before any processing delay occurs since the value is updated whenever a preemption takes place. If the value is referred to later, it should be assigned to the entity's attribute or to a local variable.

If several entities were preempted at a location, the value returned by the function to each entity will be taken from whichever entity has the largest time remaining in the WAIT statement.

When no units are specified in the parentheses in this function, it returns the default time unit specified in the General Information dialog.

Valid In

Operation logic defined as a preemptive process.

Example

You may preempt an entity called Gear while it processes at a location called Lathe1. When you preempt the Gear, it should go to a location called Wait_Area where it waits to return to Lathe1. When it returns to Lathe1, the lathe should continue processing the Gear from where it left off when you preempted the Gear. For example, if Lathe1 must process the Gear for a total of 10 minutes, but it only processes for 8 minutes before you preempt the Gear, Lathe1 should process it for only 2 additional minutes when it returns to Lathe1. To do this, we assign the remaining process time using TIMELEFT() to an attribute, Att1. We also check Att1 at Lathe1 to determine if it is greater than 0 to know whether the Gear was executing the process for the first time or as a preempted entity. Processing should be as follows:

Process Table

	Location	Operation (min)

Gear	Lathe1	IF Att1=0 THEN WAIT 10
		ELSE WAIT Att1

Gear *	Lathe1	Att1=TIMELEFT()
Gear	Wait_Area	

* Preemptive Process Record

Routing Table

	Output	Destination	Rule	Move Logic
1	Gear	Lathe2	FIRST 1	
1	Gear	Wait_Area	FIRST 1	
1	Gear	Lathe1, 99	FIRST 1	

See Also

"Preemption Process Logic" on page 300.

TimesUsed()

General System Function

Syntax samples

TIMESUSED(<resource>)

IF TIMESUSED(Res1) > 5 then USE Res2 for 10

Description

Returns the number of times a resource has been used.

Valid In

Any logic and any field evaluated after translation. For a list of fields evaluated after translation, see the “Appendix A” on page 587.

Components

<resource>

The name of the resource to examine. RES() may be used here.

Example

In the example below, when an EntA arrives at Loc1, it will only use the resource Res1 if the resource has been used five or fewer times. If the resource has been used more than five times, EntA will use Res2 instead.

Process Table

	Location	Operation (min)

EntA	Loc1	IF TIMESUSED (Res1)> 5
		THEN USE Res2 FOR 10
		ELSE USE Res1 FOR 10

Routing Table

Blk	Output	Destination	Rule	Move Logic

See Also

UNITS().

Trace

General Action Statement

Syntax samples

TRACE {<message>}{STEP or CONT or OFF or CLOSE}

TRACE "Begin Test for Resource A"

TRACE CONT

TRACE CLOSE

Description

Turns tracing on and off. Trace listings will appear in a separate window on the screen. Use trace to follow the logical flow of a model.

Valid In

Any logic.

Components

<message>

The message to appear in the trace listing when the TRACE statement is encountered. The message can be any string expression.

STEP

Makes ProModel wait for the user to click the left mouse button to execute the next statement or trace continuously while the right mouse button is held down. TRACE statements default to step.

CONT

Steps continuously without user intervention. Clicking the right mouse button will step through the logic.

OFF

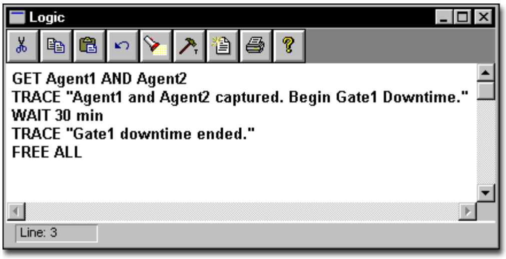
Turns tracing off but does not close the trace listing.

CLOSE

Turns tracing off and closes the trace listing.

Example

In this example, a message will appear in the trace listing whenever Agent1 and Agent2 have been captured and the downtime for Gate1 begins. Another message will appear in the trace listing at the end of the downtime.



```
GET Agent1 AND Agent2
TRACE "Agent1 and Agent2 captured. Begin Gate1 Downtime."
WAIT 30 min
TRACE "Gate1 downtime ended."
FREE ALL
```

See Also

DEBUG.

Trunc()

Type Conversion Function

Syntax samples

```
TRUNC(<expression>)
Integer1=TRUNC(3.9)
```

Description

Returns a real expression truncated to an integer. Any digits to the right of the decimal place will be removed. When necessary, ProModel automatically converts real values to integers by truncating them. For more information about ProModel automatically converting between reals and integers, see “Converting Between Numeric Types” on page 407.

Valid In

Any expression. This function returns an integer.

Components

<expression>

The expression to be truncated.

See Also

ROUND().

Ungroup

Entity-Related Operation Statement

Syntax samples

UNGROUP {LIFO}

UNGROUP

UNGROUP LIFO

Description

Separates entities that were grouped with the GROUP statement. Each of the resulting entities searches ahead in the process list and then from the beginning of the list until a process is found that has been defined for that entity type at the current location. The first entity processed from the group takes any resources the group owns. If a grouped entity has members that are also grouped entities, only the top level group is ungrouped with an UNGROUP statement. An additional UNGROUP will ungroup any member groups.

Valid In

The operation column of process edit tables only. You may not use UNGROUP on conveyors nor at the end of a queue. UNGROUP may not be used in combination with COMBINE, CREATE, UNGROUP, LOAD, UNLOAD, SPLIT AS or other UNGROUP statements in a processing logic. It may follow a GROUP statement no more than once in the same processing logic to allow batch processing.

Please note

If you are trying to ungroup an entity that has never been grouped, ProModel ignores the UNGROUP statement.

Components

LIFO

Last In, First Out. Starts the ungrouped entities processing from last to first, rather than from first to last. If this option is not specified, the ungrouped entities will be processed FIFO, or First In, First Out.

Explicit Entity Actions

With an UNGROUP, ProModel dissolves the temporary shell and divides costs among the ungrouped entities (ungrouped entities may include smaller clusters of grouped entities).

Example

The example below is the continuation of the GROUP statement example where EntA, EntB and EntC were grouped to form Grp_A. Now the entities are ungrouped with all of their original properties. (See the GROUP statement example.)

Process Table

	Location	Operation (min)
Grp_A	Loc3	UNGROUP
EntA	Loc3	USE Res1 FOR 2 min
EntB	Loc3	USE Res1 FOR 2 min
EntC	Loc3	USE Res1 FOR 2 min

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc4	FIRST 1	MOVE FOR 2
1	EntB	Loc5	FIRST 1	MOVE FOR 2
1	EntC	Loc6	FIRST 1	MOVE FOR 2

See Also

GROUP, LOAD, JOIN, COMBINE, and SPLIT AS. See “Attributes” on page 225 for more information.

Units()

General System Function

Syntax samples

UNITS(<location> or <resource>)

PAUSE "There are" $ UNITS(Res1) $ "Res1's in the system."

Description

Returns the total units of a location or resource.

Valid In

Any logic and any field except those evaluated before translation. For a list of fields evaluated before translation, see "Appendix A" on page 587.

Components

<location>

The name of the location to examine. You may substitute LOC() for the name of a location.

<resource>

The name of the resource to examine. You may substitute RES() for the name of a resource.

See Also

FREEUNITS() and RESQTY().

Unload

Entity-Related Operation Statement

Syntax samples

UNLOAD <expression> {IFF <Boolean expression>}

UNLOAD 5

UNLOAD 5 IFF Entity() = EntA

Description

Unloads a certain quantity of entities, or a certain quantity of those entities depending on a condition. Use UNLOAD to unload entities from a carrier entity that was previously loaded with LOAD. The unloaded entities are processed ahead of the entity which unloaded them. Each unloaded entity searches ahead in the process list, and then from the beginning of the list, until a process is found for that entity type at that location.

Valid In

The operation column of process edit tables only. UNLOAD is not valid at conveyors, after routing, or at the end of a queue. You may not use UNLOAD in combination with COMBINE, CREATE, GROUP, UNGROUP, and SPLIT AS or other UNLOAD in the same process logic. If the process contains LOAD statements, UNLOAD can only appear once after all of them.

Components

<expression>

The number of entities to unload. A value of zero is ignored and a negative value produces an error. If the quantity specified for unloading is greater than the number of entities that have been loaded, the extra quantity is ignored. This expression is evaluated every time the statement is encountered.

IFF <Boolean expression>

This option allows the UNLOAD command to be conditional. Any attributes, entity functions, and location functions apply to the entity to be unloaded, not to the current entity. This technique allows only entities with certain properties to be unloaded from the current entity. To use attributes, entity functions, and location functions that apply to the current entity, assign the desired value to a local variable and use the local variable in the Boolean expression.

Explicit Entity Actions

UNLOAD divides up costs and copies statistical information for duration of loading to each entity.

Example

The following example is a continuation of the LOAD statement example and shows how the loaded entity (Truck) is unloaded, resulting in the original Truck and the boxes that were loaded onto it. Boxes continue to the next location while Truck is returned to its starting location, Factory. (See the Load statement example.)

Process Table

	Location	Operation (min)
Box	Shipping	WAIT 2 min
Truck	MfgSite	
Truck	Dock	LOAD Attr1 IN 2 Hr
Truck	NewYork	WAIT T(20,30,60) UNLOAD 5
Box	NewYork	
Truck	Chicago	WAIT T(20,30,60) UNLOAD 5
Box	Chicago	
Truck	Boston	WAIT T(20,30,60) UNLOAD 5
Box	Boston	

Routing Table

	Output	Destination	Rule	Move Logic
1	Box	Dock	LOAD 1	MOVE FOR 45 sec
1	Truck	Dock	FIRST 1	MOVE FOR 10 min
1	Truck	NewYork	FIRST 1	MOVE FOR 24 Hr
	Truck	Chicago	FIRST	MOVE FOR 12 Hr
	Truck	Boston	FIRST	MOVE FOR 28 Hr
1	Truck	MfgSite	FIRST 1	MOVE FOR 24 Hr
1	Box	NY_Recv	FIRST 1	MOVE FOR 5 min
1	Truck	MfgSite	FIRST 1	MOVE FOR 12 Hr
1	Box	Chi_Recv	FIRST 1	MOVE FOR 5 min
1	Truck	MfgSite	FIRST 1	MOVE FOR 28 Hr
1	Box	Bos_Recv	FIRST 1	MOVE FOR 5 min

See Also

LOAD, COMBINE, JOIN, GROUP, and UNGROUP. Also see “Attributes” on page 225 for more information.

Use

Resource-Related Operation Statement

Syntax samples

```
USE {<quantity>} <resource> {,<priority>}
      FOR <time> {AND or OR {<quantity>}
      <resource> {,<priority>}  FOR <time>... }
```

```
USE 2 Res2, 5 FOR 4:23:03
```

```
USE 2 Res1 FOR 2.0 min OR 3 Res2 FOR 1.5 min
```

```
USE Res1, 3 FOR 1 hr AND (Res2 FOR 5 OR Res3
FOR 5)
```

```
USE Oper_Attr RES(Type_Attr) FOR Time_Var1
Hr.
```

Description

Captures a resource or combination of resources as each resource becomes available. Once the resource has been captured it is used for the specified amount of time, and then freed when the specified duration is over. If the entity already possesses one of the specified resources from a previous GET, JOINTLY GET, or MOVE WITH statement, the entity will still try to capture an additional unit of that resource.

Please note

If an entity uses a USE statement to capture a resource, the resource must complete its operation before you can preempt the entity. However, if the entity uses a GET, WAIT, FREE sequence, you may preempt the entity during the WAIT portion of the logic.

Valid In

Location processing logic, downtime logic, and move logic.

Components

<quantity>

The number of resources to get. ProModel ignores a value of zero and values less than zero return an error. ProModel evaluates and truncates this numeric expression every time it encounters the USE statement.

<resource>

The name of the resource to USE. You can substitute RES() for the resource name.

<priority>

When multiple entities request a resource, ProModel fills the requests in order of priority. This expression should be a number between 0 and 999.

<time>

The length of time the entity will tie up the resource. ProModel evaluates this expression whenever it encounters the statement. If you specify no time unit, ProModel applies the default time unit specified in the General Information dialog.

Example

This simple example shows how Clients arriving at location Reception must USE the Secretary for some amount of time according to the user-defined distribution, Dist1. Clients are then sent to a waiting area until the desired next location becomes available.

Process Table

	Location	Operation (min)
Client	Reception	USE Secretary FOR Dist1()
Client	Waiting	

Routing Table

	Output	Destination	Rule	Move Logic
1	Client	Waiting	FIRST 1	MOVE FOR 30 sec
1	Client	Loan	.400 1	MOVE FOR 1
	Client	Auditor	.350 1	MOVE FOR 1
	Client	Service	.250 1	MOVE FOR 1

See Also

GET, JOINTLY GET, and MOVE WITH.

Variable()

General System Function

Syntax samples

VARIABLE(<numeric expression>)

VARIABLE(Attr) = 124

VARIABLE(x) = VARIABLE(y)

N=VARIABLE(x) + VARIABLE(y) - 1

Description

Converts a name-index number or integer to a variable name. Use this function when a numeric expression uses a variable whose name-index number is stored in an attribute, array, or variable.

Valid In

Any logic.

Please note

You cannot use VARIABLE() in a PROMPT, INC, DEC, or READ statement.

Components

<numeric expression>

The name index number for a variable. You can determine the name-index number associated with a particular variable by the position of the variable record in the Variables module.

Example

.In the example below, parts of different types arrive at location In_Queue. Each entity type has a unique value for Attr1 that corresponds to the name-index number of a variable in the Variables module. Once parts arrive at In_Queue, they increment the variable specific to that entity type before routing to the location Process_Loc.

Process Table

	Location	Operation (min)
All	In_Queue	VARIABLE(Attr1) = VARIABLE(Attr1) + 1

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	All	Process_Loc	FIRST1	MOVE 1

See Also

LOC(), ENT(), and RES().

View

General Action Statement

Syntax samples

VIEW "view name"

VIEW "Cell5"

VIEW "View10"

Description

Use this statement to change the view in the Layout window from within your logic. Once the view has been defined from the View menu in main menu, you can use it in the logic.

Valid In

All logic.

Components

<view name>

The name of the view defined in the Views dialog. Enclose the name in quotation marks.

Example

You are giving a presentation on the use of simulation for airport design. Two hours into the model run, you want to zoom in on the baggage area to show the activity there. Three hours into the simulation, you want to zoom out to show the entire airport..You are giving a presentation to management on the factory floor using simulation. Two hours into the simulation, you want to zoom in on a particular cell in the factory to show the activity there. Three hours into the simulation, you want to zoom out to show the entire factory.

To do this, define two views called Cell1 and Factory using the Views editor on the View menu. Define an independent subroutine and call it in the initialization logic using the ACTIVATE statement. Enter the following logic in the subroutine:

```
INT X=1
WHILE X=1 DO
BEGIN
IF CLOCK(hr) = 2 THEN VIEW "Cell1"
IF CLOCK(hr) = 3 THEN VIEW "Factory"
WAIT 1 hr

END
```

See Also

"Commands" on page 90.

Wait

Entity-Related Operation Statement

Syntax samples

WAIT <time expression>

WAIT 3 min

WAIT 0

WAIT 2.5 + CleanupTime

WAIT N(8,.5) + 3 sec

Description

Simulates the time it takes to process an entity. WAIT delays further processing of the entity until the specified time has elapsed. The rest of the model continues to process while an entity waits. If the expression evaluates to zero, the current entity will not finish processing until all other processes scheduled for that moment in the simulation have finished.

Please note

You may use the "^" symbol in place of a "WAIT" statement.

Valid In

Location processing, downtime, and move logic. Independent subroutines may also use WAIT statements which function as timers. (For more information about Independent subroutines, see "Subroutines" on page 246.)

Components

<time expression>

The length of the WAIT. This expression is evaluated whenever the statement is encountered. If no time unit is specified, the default time unit specified in the General Information Dialog is applied.

Example

In the following example, customers arrive at a fast-food restaurant and place their orders at location Counter. How long customers wait depends on the type of meal ordered. If they order meal one, two, or three, they wait only 2 minutes. Otherwise they wait 5 minutes.The example below shows WAIT statements used in an IF...THEN...ELSE expression. If the value of EntA's attribute, Attr1, is greater than zero, the time delay is 5.0 minutes. Otherwise the delay is 2.5 minutes.

Process Table

	Location	Operation (min)

EntA	Loc1	IF Attr1>0 THEN
		   WAIT 5 min
		ELSE
		   WAIT 2.5 min

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	EntA	Loc2	FIRST 1	MOVE FOR 10

See Also

WAIT UNTIL can stop additional processing until a condition is true.

Wait Until

Entity And Resource-Related Operation Statement

Syntax samples

WAIT UNTIL <Boolean expression>

WAIT UNTIL Var1 > 3

WAIT UNTIL Var1 < Attr3 AND Var2 >= 5

Description

Delays processing of the current logic until the Boolean expression is true. The rest of the model continues to process during the delay. Note that if the expression is initially evaluated to be false, it is only reevaluated when a location attribute, variable, or array element in the expression changes. Multiple entities waiting on the same condition are released one at a time. This allows a released entity to change a variable value that will prevent the other waiting entities from being released.

Valid In

Node entry and node exit logic delays processing for resources, and location processing logic delays processing for entities. Independent Subroutines. (See “Subroutines” on page 246.)

Components

<Boolean expression>

The condition that must be satisfied to continue processing the entity or resource. Elements of this expression are limited to location attributes, variables, and array elements.

Example

The example below uses the WAIT UNTIL statement to group a variable quantity of entities. As each EntA arrives at Loc1, a variable (Total) is incremented to keep track of the total entities waiting at the location. The WAIT UNTIL statement causes processing of all EntA’s to halt at this point until the variable, Var1, reaches or exceeds 500. Then all of the waiting EntA’s are GROUPed together as a BatchA.

Process Table

	Location	Operation (min)
EntA	Loc1	INC Total WAIT UNTIL Total >= 5 GROUP Total AS Batch

Routing Table

Blk	Output	Destination	Rule	Move Logic

See Also

DO...WHILE, WHILE...DO, and DO...UNTIL.

Warmup

General Action Statement

Syntax samples

```
WARMUP
WARMUP
IF thruput = 50 THEN WARMUP
```

Description

Instructs the simulation to end the warmup period by resetting all the statistics and erasing relevant time series files. Only one WARMUP statement may be used in the simulation.

Please note

If multiple WARMUP statements are encountered or a WARMUP statement is used in addition to the warmup time specified in the Run Options dialog, ProModel will generate a warning to inform you that this has occurred. Only the first warmup encountered (the statement or the Run Options setting) will actually be executed. While the statement will be encountered within the logic, the Run Options setting will be executed at the specific time indicated in the dialog.

Valid In

Any logic.

Example

Suppose you want to base the warmup period on 2,000 entities being processed rather than on a lapse of time. You could increment a variable (e.g., Total_Processed) whenever an entity exited the system. Enter the following logic in an independent subroutine activated from the initialization logic:

```
WAIT UNTIL Total_Processed = 2000
WARMUP
```

See Also

See “Simulation Options” on page 348 for more information on warm-up periods.

While...Do

General Control Statement

Syntax samples

WHILE <Boolean expression> DO <statement block>

WHILE Array1(n) <> 10 DO INC n

```
WHILE FREECAP(Loc1) > 5 DO
    BEGIN
        INC Var2, 5
        WAIT 5 sec
    END
```

Description

Repeats a statement or statement block continuously while a condition remains true. WHILE...DO is an entry-condition loop, meaning that the loop will not be executed once unless the Boolean expression is true.

Valid In

Any logic.

Components

<Boolean expression>

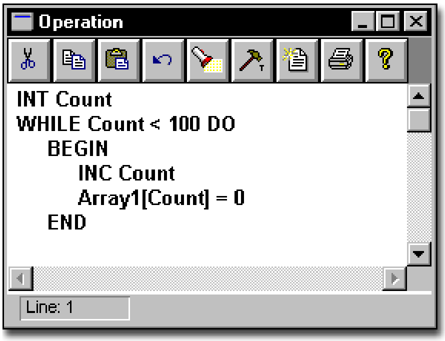
As long as this expression is TRUE, the loop will continue. This expression is evaluated for each iteration of the loop.

<statement block>

The statement or block of statements to execute.

Example

The example below shows the arrival logic window for an entity that arrives every 40 hours to reset the values of Array1 elements 1 through 100 to 0.



```
INT Count
WHILE Count < 100 DO
   BEGIN
      INC Count
      Array1[Count] = 0
   END
```

See Also

BEGIN, END, DO...WHILE, and DO...UNTIL.

Write

General Operation Statement

Syntax samples

WRITE <file ID>, <string or numeric expression> {,<maximum digits before decimal>, <digits after decimal>}

Description

Writes information to a general write file. The next item written to the file will appear immediately after this item. WRITE always appends to the file unless the file is RESET. This holds true for multiple replications as well as single, independent runs. Any file that is written to with WRITE automatically becomes a text file and will have an end of file marker attached automatically to the end when it is closed. For more flexible WRITE capability, use XWRITE.

WRITE and WRITELINE automatically separate values by commas

Valid In

Any logic.

Components

<file ID>

The name of the file as previously defined in the External Files Editor.

<string or numeric expression>

The string or numeric expression to be written to the file. In the output file, quotes will automatically be added to string expressions so that most spreadsheet programs can easily read the file.

<maximum digits before decimal>

The maximum number of digits before the decimal. This value is used to line up any numeric values into columns on the decimal point. This value may be any numeric expression.

<digits after decimal>

The number of spaces to save after the decimal point. Use this option to line up any labels appearing after numbers.

Example

The following example uses both WRITE and WRITELINE to record the time when EntA completes processing at Loc1 in a general write file called Rpt.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT N(7.3,.4) WRITE Rpt,"EntA Complete at:" WRITELINE, Rpt CLOCK(min),3,2

Routing Table

Blk	Output	Destination	Rule	Move Logic

See Also

XWRITE, WRITELINE, RESET, READ, and FORMAT(). Also see “External Files” on page 262.

Please note

The sum of the maximum digits before and after the decimal must be less than 20.

WriteLine

General Operation Statement

Syntax samples

WRITELINE <file ID>, <string or numeric expression>{,<maximum digits before decimal>, <digits after decimal>}

Description

Writes information to a general write file and starts a new line. WRITELINE always appends to the file unless the file is RESET. Any file that is written to with WRITELINE automatically becomes a text file and will have an end of file marker attached to the end when it is closed.

WRITE and WRITELINE automatically separate values by commas

Valid In

Any logic.

Components

<file ID>

The name of the file as previously defined in the External Files Editor.

<string or numeric expression>

The string or numeric expression to be written to the file. In the output file, quotes will automatically be added to string expressions so that most spreadsheet programs can easily read the file.

<maximum digits before decimal>

When writing real numbers, the maximum number of digits before the decimal. Use this value to line numbers on different lines up on the decimal.

<digits after decimal>

When writing real numbers, the maximum number of digits after the decimal.

Example

The following example uses both WRITE and WRITELINE to record the time when EntA completes processing at Loc1 in a general write file called Rpt.

Process Table

	Location	Operation (min)
EntA	Loc1	WAIT N(7.3,.4) WRITE Rpt, "EntA Complete at:" WRITELINE Rpt, CLOCK(min),3,2

Routing Table

Blk	Output	Destination	Rule	Move Logic

See Also

WRITE, RESET, XWRITE, READ, and FORMAT(). Also see “External Files” on page 262.

Please note

The sum of the maximum digits before and after the decimal must be less than 20.

Xsub()

External Subroutine Call

Syntax samples

XSUB(<file ID>, <ordinal function number> or <function name> {, <parameter1>, <parameter2>...})

XSUB(Interface,1, 5)

XSUB(LogDLL, "_Log_B_of_X",10,5)

Description

Calls an external subroutine inside a DLL file. XSUB() is perhaps the most powerful statement in ProModel, because by using it the user can access the entire functionality of any 32-bit Windows programming language such as C, C++, Delphi, or Visual Basic. XSUB() can be used for sophisticated file IO and to make simulations interactive. In fact, subroutines called with XSUB() can do anything that the language they were written in allows. Because of its power, however, XSUB() should be used with caution. When called, the simulation is suspended until the external subroutine finishes execution.

The subroutine inside the DLL must have been compiled as exportable by a Windows 32-bit compiler and have a return type of IEEE format double real. XSUB() will copy the parameters following the function name to a block of memory, then pass the function a pointer to that block of memory.

The function can take only one parameter: a pointer to void. But the function may access any number of parameters through structure overlaying. The function should define a structure to match the type and order of the parameters, and assign the pointer just passed to a pointer to that type of structure. The parameters can then be used through structure overlaying. Integers are passed as four byte values and reals are passed as eight byte IEEE values.

Please note

For Windows Programmers Only *The handle to the run-time simulation frame window will be the last parameter passed. Most subroutines can completely ignore this parameter, but it will be needed if the subroutine displays a window.*

Valid In

Any expression or logic.

Components

<file ID>

The file ID assigned to an external DLL file as defined in the External Files editor. This file should be a 32-bit Windows DLL file.

<ordinal function number>

The ordinal number of the function inside the DLL. This function must be exportable. When DLL's are compiled, every exported function inside them is numbered. The individual functions can then be accessed by calling the program by number. This field may be an expression that evaluates to an ordinal function number that is valid inside the DLL. Use an ordinal function number or the function name.

<function name>

The name of the function inside the DLL. This function must be exportable. Note that when most compilers compile DLL's, they adjust the name of the functions inside them. The function name inside the XSUB statement must be the adjusted name, not the original name.

Most C compilers add an underscore to the function name; so a function called "Test1" would be compiled as "_Test1." For most C++ compilers, valid ProModel external function names will be mangled to "@<function name>$pv." Different compilers will vary, however, so the user should be aware of the particular compiler's quirks.

<parameters>

The parameters to pass to the function. These may be any variable, number, or expression. They are only limited by the type of field or logic that uses the XSUB function. Each parameter should be separated by a comma. See above for how the external subroutine will access these parameters.

Example

An external function written to the ProModel specification, called Log_B_of_X and written in C, returns the log of a value to a variable base. The function, reproduced below, has been compiled into the DLL, "XSUB.DLL." The function itself is reproduced on the left below, and the source code can be found in the file "XSUB.CPP."

The ProModel logic statements assign the base five logarithm of the real variable R1 to the real variable R2. Each statement assumes that the file XSUB.DLL has been assigned the identifier "Log" in the External Files Editor. The first statement accesses the function as if the DLL had been compiled in C++ by using the mangled function name. The second statement accesses the function as if it had been compiled in C by using the C adjusted function name. The third statement accesses the function using the ordinal function number.

ProModel Logic

```
R2 = XSUB(Log, "@Log_B_X$pv",5.0,R1)
R2 = XSUB(Log, "_Log_B_X",5.0,R1)
R2 = XSUB(Log, 2, 5.0, R1)
```

XSUB.CPP

```
struct TEST_SUB_PARAMS

{
double base;
double x;
HWND hWndFrame;
};

extern "C" //This makes this a C function rather
than a C++ function
{
/* On compile, Borland and Microsoft add a
leading underscore  to a 'C' name! */
double _export Log_B_of_X(void *p)
{

//Parameters come in a structure pointed to by p;
TEST_SUB_PARAMS * params;
params = (TEST_SUB_PARAMS*) p;
MessageBox (GetTopWindow (params->hWnd-
Frame),
"Executing Log_B_of_X function.", "XSUB",
MB_OK);
return log (params->x) / log (params->base);
}
}
```

See Also

"Subroutines" on page 246 (normal subroutines are less powerful, but much easier to create and use).

Xwrite

General Operation Statement

Syntax samples

XWRITE <file ID>, <string or numeric expression>

Description

Writes information to a general write file in any format the user chooses. XWRITE is for writing user-formatted files, while WRITE and WRITELINE are for writing special formatted text files. XWRITE always appends to a file unless the file is RESET. Note that any time a WRITE or WRITELINE writes to a file, the file will automatically be a text file. No end of file marker is appended to files written only with XWRITE. In subsequent replications, additional items are appended to the end of the file unless the file is RESET. ProModel does not format the string expression, although you can use the FORMAT statement to manually format data. To add an end of file marker to a user-formatted file, use XWRITE CHAR(26).

Valid In

Any logic.

Components

<file ID>

The name of the file as previously defined in the External Files Editor.

<string or numeric expression>

The string expression to be written to the file.

Example

The following example uses XWRITE to record the time each Box completes processing at location Ship.

Process Table

	Location	Operation (min)
Box	Ship	WAIT N(7.3,.4) XWRITE Rpt, “Box Shipped at:” $ FORMAT(CLOCK(min),3,2)

Routing Table

Blk	Output	Destination	Rule	Move Logic
1	Dock	Doc	FIRST 1	MOVE FOR 5

See Also

WRITE, WRITELINE, and RESET. Also see “External Files” on page 262.

Appendix A

ProModel evaluates various fields and logic at different times during a simulation run. Some evaluate only once when the model prepares to run a simulation, called the translation time. ProModel evaluates others as necessary throughout the simulation. Below is a list of all expressions and statements by category, and a table telling which fields evaluate at translation and which fields ProModel evaluates continuously during a simulation. The table also tells where you may use an expression or statement.

Expression and Statement Groups

1. General Expression

Numbers, Variables, Math Functions, Functions Table, Distribution Functions, XSUB(), and Name Functions (i.e., RES(), ENT(), LOC()).

2. Arrays

Arrays of any dimension.

3. Location Attributes

Referencing a location attribute in any expression or assigning to a location attribute.

4. Entity-Specific System Functions and Attributes

Entity(), GroupQty(), ResQty().

5. General System Functions

Cap(), CalDay(), CalDom(), CalHour(), CalMin(),CalMonth(), CalYear(), Clock(), Contents(), DownQty(), Entries(), FreeCap(), FreeUnits(), OwnedResource(), TimesUsed(), Units(), and Variable().

6. Location-Specific System Function

Location().

7. Resource-Specific System Function

Resource().

8. Node Logic-Specific System Functions

Last(), Next(). (The Last function is valid only in Node Entry Logic and the Next function is valid only in Node Exit Logic.)

9. Downtime-Specific System Function

DTDelay().

10. Shift-Specific System Functions

ForLocation(), ForResource().

11. Preemption Logic-Specific System Functions

Preemptor(), TimeLeft().

12. Off-Shift & Break Logic-Specific System Functions & Statements

DTLeft(), Priority, and Skip.

13. Cost Functions & Statements

GetCost(), GetResRate(), IncEntCost, IncLocCost, and IncResCost.

14. General Statements: (Run-Time Only)

Activate, Animate, Assignment, Begin End, Break, BreakBLK, Close, Comment(#, //, /*...*/), Debug, Dec, Display, Do...Until, Do...While, Goto, If...Then/If...Then...Else, Inc, Int, Local Variable, Log, MapArr, Order, Pause, Prompt, Read, Real, Reset, Reset Stats, Return, Report, Send, SetRate, Stop, String Expressions, Sound, Trace, View, WarmUp, While...Do, and Write/Writeline/XWrite. (You may not use the Log statement in Initialization or Termination Logic.)

15. Operation Statements (Group 1)

Accum, Combine, Create, Group, Join, Load, Match, Move, Rename, Route, Split As, Ungroup, and Unload.

16. Operation Statements (Group 2)

Free, Get, Jointly Get, and Use.

17. Operation Statements (Group 3)

Wait.

18. Move Logic-Specific Statements

Move For, Move On, and Move With.

19. External Spreadsheet File

Entity Location Expression File.

20. Graphic Statement

Graphic.

21. Wait Until Statement

Wait until.

Macros and Subroutines

A macro may be used in any expression field, but the macro may only contain expressions which return a value (e.g., Entries(LOC1), U(5,1)). When a macro is used in a logic field, the macro may include any logic element that is valid in that logic field. A subroutine may also be used in an expression field provided that the Return statement is used to return a value to the expression field. When a subroutine is used in a logic field, the subroutine may include any logic element that is valid in that logic field.

Valid Expression and Statement Groups by Field

When running a simulation, expressions and statements are either evaluated 1) once at translation (before initialization logic and before any events are created) or 2) continuously during the simulation run. The following chart categorizes the edit fields in this manner and shows the expression and statement groups that can be used in each field. Fields are classified as either expression fields or logic fields. Fields not listed on this chart are either menu fields or expression fields that accept only numbers.

Fields Evaluated Only at Translation

Field Name (evaluated at translation)	Field Type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Variables																						
Initial Value	Exp	•																				
Function Table																						
Dependent Value	Exp	•																				
Independent Value	Exp	•																				
Simulation Options																						
Warm-up Hours	Exp	•																				
Run Hours	Exp	•																				
Replications	Exp	•																				
Interval Length	Exp	•																				
Path Networks																						
Node Capacity	Exp	•																				
Segment Distance	Exp	•																				
Speed Factor	Exp	•																				
Resources																						
Resource Units	Exp	•																				
Locations																						
Location Capacity	Exp	•																				
Conveyor																						
Length	Exp	•																				
Speed	Exp	•																				
Queue																						
Queue Length	Exp	•																				

Field Name (evaluated at translation)	Field Type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Cycle Tables																						
Time (Hours)	Exp	•																				
%	Exp	•																				
Qty	Exp	•																				
Shifts																						
Shift Start Time	Exp	•																				
Entities																						
Entity Length	Exp	•																				
Entity Width	Exp	•																				

Fields Evaluated During Simulation

Field Names (during simulation)	Field Type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Logic																						
Initialization Logic	Logic	•	•			•									•							
Termination Logic	Logic	•	•			•									•							
Entities																						
Speed	Exp	•	•	•	•	•	•													•		
Path Networks																						
Traveling Time per Path Segment	Exp	•	•		•	•																
Resources																						
Acceleration	Exp	•	•		•	•																
Deceleration	Exp	•	•		•	•																
Empty Load Speed	Exp	•	•			•																
Full Load Speed	Exp	•	•		•	•																
Pickup Time	Exp	•	•	•	•	•	•													•		
Deposit Time	Exp	•	•	•	•	•	•													•		
Node Entry Logic	Logic	•	•			•		•	•						•						•	•
Node Exit Logic	Logic	•	•			•		•	•						•						•	•
Location Clock DT																						
First Occurrence	Exp	•	•			•																
Frequency	Exp	•	•			•																
Priority	Exp	•	•			•																
Clock DT Logic	Logic	•	•	•		•	•			•					•		•	•				
Location Entry DT																						
First Occurrence	Exp	•	•			•																
Frequency	Exp	•	•			•																
Entry DT Logic	Logic	•	•	•		•	•			•					•		•	•				
Location Usage DT																						
First Occurrence	Exp	•	•			•																
Frequency	Exp	•	•			•																
Priority	Exp	•	•			•																
Usage DT Logic	Logic	•	•	•		•	•			•					•		•	•				

Field Names (during simulation)	Field Type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Location Setup DT																						
Setup DT Logic	Logic	•	•	•	•	•	•			•					•		•	•		•		
Resource Clock DT																						
First Occurrence	Exp	•	•			•																
Frequency	Exp	•	•			•																
Priority	Exp	•	•			•																
Clock DT Logic	Logic	•	•			•		•		•					•		•	•			•	
Resource Usage DT																						
First Occurrence	Exp	•	•			•																
Frequency	Exp	•	•			•																
Priority	Exp	•	•			•																
Usage DT Logic	Logic	•	•			•		•		•					•		•	•			•	
Operation																						
Operation Logic	Logic	•	•	•	•	•	•							•	•	•	•	•		•	•	•
Preemption																						
Preemption Logic	Logic	•	•	•	•	•	•					•			•	•	•	•		•	•	•
Routing																						
Priority for Destination	Exp	•	•	•	•	•	•													•		
Destination	Exp	•	•	•	•	•	•													•		
Entity Output Quantity	Exp	•	•	•	•	•	•													•		
User Condition Rule	Exp	•	•	•	•	•	•													•		
Move Logic	Logic	•	•	•	•	•	•							•	•		•	•	•	•	•	•
Arrivals																						
First Occurrence	Exp	•	•			•																
Frequency	Exp	•	•			•																
Occurrences	Exp	•	•			•																
Qty of Each Arrival	Exp	•	•			•																
Arrival Logic	Logic	•	•	•	•	•	•								•					•	•	
Shift Assignments																						
Priorities	Exp	•	•																			
Pre-Off Shift Logic	Logic	•	•			•		•			•				•			•				•
Off Shift Logic	Logic	•	•			•		•			•		•		•		•	•				•

Field Names (during simulation)	Field Type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Pre-Break Logic	Logic	•	•			•		•			•				•			•				•
Break Logic	Logic	•	•			•		•			•		•		•		•	•				•
Subroutines																						
Interactive Subroutines	Logic	•	•			•									•			•				•
Debugger																						
Debug Condition	Exp	•	•	•	•	•	•															

Program Defaults

This section discusses and lists the default settings when you first run ProModel.

As in most software packages, ProModel makes certain assumptions regarding the most useful settings for the program. While your models may require different settings, an effort has been made to select the most common and helpful default settings.

Since ProModel can be used for a wide variety of applications, you may require different settings. As your models differ, so will the need to change these settings from their default. However, when ProModel is first launched and no INI file exists for it, the defaults in the following table apply. Some defaults involve edit field values, some are check box settings, etc.

Many defaults can be changed by selecting one of the Model Default options under the Tools menu. Other defaults can only be changed in the Build modules. Still others can only be changed by directly editing the .INI file.

After ProModel is first launched, the promod.ini(initialization) file is created in the Windows directory. To change the .INI file defaults, you must edit the it using an ASCII text editor (e.g., Windows Notepad). DO NOT attempt to edit the .INI file unless you are fully comfortable doing so and only after you make a backup copy of the original .INI file.

Default Values

Graphic library	promod.glb
Locations	
Conveyor/queue style	roller
Conveyor/queue width (feet)	3
Conveyor speed (fpm)	60
Conveyor accumulating	accum
Conveyor entity orientation	lengthwise
Location statistics	time series
Entities	
Entity speed	150 fpm (50 mpm)
Entity statistics	time series
Path Networks	
Time/speed	Speed
Path color	blue/yellow
Resources	
Speed	150 fpm (50 mpm)
Entity search	Longest Waiting
Resource search	Closest resource
Resource statistics	By unit
Processing	
Snap to border	off
Variable statistics	time series

Long build menu	on
Confirm record delete	on
Recalculate path lengths when adjusted	on

View Menu	
Snap to grid	off
Show grid	off
Grid size (VGA full layout)	90 grid units
Scale - distance	1 ft/grid unit
Scale - time	.01 min/grid unit
Background color	lt. gray
Edit table font	Arial 8 Regular
Edit table color	Gray

Directories	
Models	c:\ProModel\models
Glib	c:\ProModel\glib
Output	c:\ProModel\output

Graphic Editor	
Text alignment	center
Text frame type	transparent
Text frame shape	rectangle
Text font	Arial 8 Bold
Text color	black
Text frame border color	black
Text frame fill color	white
Variables	
Digit font	Arial 8 Bold
Digit color	yellow
Digit frame type	recessed
Digit frame shape	rectangle
Digit frame border color	black
Digit frame fill color	royal blue
Conveyors/Queues	
Queue border color	black
Queue fill color	dark gray
Gauges/Tanks	
Gauge/tank fill color	maroon
Gauge/tank border color	black
Gauge/tank empty color	white
Gauge/tank - no border	checked
Gauge/tank - show scale	checked
Other	
Autosave (in minutes)	10
Print layout - print options dialog - bkgrnd color	unchecked

ActiveX Objects

ProModel's ActiveX Automation capability allows you to externally create, view, and edit model elements such as locations, entities, and variables. Using Visual Basic (or any other ActiveX-enabled language), you can add capabilities to ProModel including:

- Customized user interface with table inputs
- Custom-designed parameter screens
- Automatic model creation from external data sources (e.g., Excel spreadsheets, databases, or ASCII text files)
- Software execution from another application

You can also use ProModel's ActiveX capability to access any of the following actions and tables (any table not mentioned in the following list is not currently ActiveX enabled).

More information on ActiveX Automation can be found in the ActiveX User Guide, which is located in the documentation folder within the ProModel directory.

Program Operations

- Start program
- Quit program
- Pause simulation model
- Query status of model (e.g., running, paused, etc.)
- Enter Run-time Interface parameters from external data source
- Access any data field provided in output database

Location Table

- Name
- Capacity
- Units
- Stats
- Incoming selection rule
- Incoming rule, max attribute
- Incoming rule, min attribute
- Queuing for Output rule
- Queuing rule, max attribute
- Queuing rule, min attribute
- Multi-unit selection rule
- Notes
- Operation rate (cost)
- Time units (cost)

Clock Downtime Subtable

- Frequency
- First time
- Priority
- Scheduled
- Logic
- Disable

Entry Downtime Subtable

- Frequency
- First time
- Logic
- Disable

Usage Downtime Subtable

- Frequency
- First time
- Priority
- Logic
- Disable

Setup Downtime Subtable

- Entity
- Prior entity
- Logic
- Disable

Entity Table

- Name
- Speed
- Stats
- Notes
- Initial cost

Resource Table

- Name
- Units
- Stats
- Resource search rule
- Entity search rule
- Entity search, min attribute
- Entity search, max attribute
- Speed empty
- Speed full
- Acceleration
- Deceleration
- Pickup time
- Drop-off time
- Return home flag
- Notes
- Network
- Home node
- Shift node
- Break node
- Operation rate (cost)
- Time units (cost)
- Cost per use

Clock Downtime Subtable

- Frequency
- First time
- Priority
- Scheduled
- List
- Node
- Logic
- Disable

Usage Downtime Subtable

- Frequency
- First time
- Priority
- List
- Node
- Logic
- Disable

Processing & Routing Tables

- Entity name
- Preemption process flag
- Location name
- Operation logic
- Routing block number
- Output entity name
- Destination name
- Destination priority
- Begin new block flag
- New entity check box
- Output quantity
- Output rule
- Output probability
- Output user condition
- Move logic

Path Networks Table

- Color
- Visible
- Name
- Type
- Basis (time or speed)

Path Segments Subtable

- From
- To
- Bi-directional
- Time

Interfaces Subtable

- Node
- Location

Mappings Subtable

- From node
- To node
- Destinations

Nodes Subtable

- Coordinates
- Name
- Capacity

Arrivals Table

- Entity name
- Location name
- Quantity of each arrival
- Qty each (cycle table name)
- First time
- Number of occurrences
- Frequency of arrivals
- Arrival logic
- Disable flag

Shift Assignments

- Locations
- Resources
- Units
- Shift files
- Start time
- Priorities
- Logic
- Disable

Scenarios

- Name
- Enabled

Model Parameters

Parameter Subtable

- Value

Attribute Table

- Attribute name
- Type
- Classification (entity or location)
- Notes

Variables Table

- Variable name
- Type
- Initial value
- Stats
- Stats basis
- Notes

Arrays Table

- Array ID
- Dimensions
- Type
- Import File
- Export File
- Notes

Macro Table

- Identification (name)
- Text
- Resource grouping

Subroutine Table

- Subroutine name
- Operation logic

User Distributions

- Distribution name
- Type
- Cumulative
- Percentage
- Value

External Files Table

- Filename
- File type
- Path
- Prompt
- Notes

General Information Dialog

- Model title
- Default time units
- Default distance units
- Graphics library file name
- Initialization logic
- Termination logic
- Model notes

Simulation Options Dialog

- Output path
- Run length type
- Warm-up period flag

Time Only

- Warm-up hours (time only)
- Run hours

Weekly

- Warm-up start (day, hr, min)
- Simulation begin (day, hr, min)
- Simulation end (day, hr, min)

Calendar Date

- Warm-up start (month, day, year, hour, minute)
- Simulation begin (month, day, year, hour, minute)
- Simulation end (month, day, year, hour, minute)
- Output report method
- Output report interval length
- Number of replications
- Clock precision
- Clock units
- Disable time series flag
- Disable animation flag
- Disable costing flag
- Pause at start flag
- Display notes flag

More on ActiveX

For detailed information regarding ActiveX objects and methods, contact:

ProModel Sales and Support Team

Phone (888) PRO-MODEL

Fax (801) 226-6046

Suggested readings

To expand your knowledge and understanding of simulation, its practices, and its applications, consider the following texts.

Harrell, Charles; Ghosh, Biman; Bowden, Royce. 2003. *Simulation Using ProModel*. 2nd Edition; McGraw-Hill, Inc.

Law, A.M. and David W. Kelton. 1991. *Simulation Modeling and Analysis*. 2nd Edition, Chapter 4; McGraw-Hill, Inc.

Lewis, P. and E. Orav. 1989. *Simulation Methodology for Statisticians, Operations Analysts, and Engineers*. Volume I; Wadsworth & Brooks.

Harrell, Charles R. and Kerim Tumay. 1995. *Simulation Made Easy.* Industrial Engineering Press.

Banks, Jerry and John S. Carson, II. 1984. *Discrete Event System Simulation*. Prentice-Hall Inc.

Heizer, Jay and Barry Render. 1988. *Production and Operations Management.* Allyn and Bacon.

Carrie, Allan. 1988. *Simulation of Manufacturing Systems*. John Wiley & Sons.

Industrial Engineering & Management Press. 1987. *Simulation: Modeling Manufacturing & Service Systems.* Institute of Industrial Engineers.

Harrell, Charles R., Robert E. Bateman, Thomas J. Gogg, Jack R.A. Mott. 1993. *System Improvement Using Simulation*. JMI Consulting Group and PROMODEL Corporation.

Glossary

Cost Statistics

Statistics collected on a cost basis (e.g., total cost and average, non-use cost). You have control of how ProModel collects these statistics. You may collect information through statements, base them on the system clock, or use a combination of both.

Counter

A counter is a location or variable graphic used to display the contents of a location or the current value of a variable during animation. A counter consists of a frame, a specification of the digit color, and the font. If the number being displayed requires more digits than the maximum specified, the counter simply expands to the left. ProModel displays real values showing only 2 decimal places. For example, a variable equal to 4.8936 would display 4.89 on the screen.

Dialog Box

A dialog box is a pop-up window used to enter information or select options. Movement from field to field in a dialog box is accomplished by clicking in the field with the left mouse button or by using the Tab key. To accept input made to a dialog box press Enter or click on the OK button. To cancel any changes made to the dialog press Esc or click on the Cancel button. For help on the dialog, press F1 or click on the Help button. If no Help button is shown, you may select Help from the main menu.

Edit Tables

An edit table is a powerful editing window used to add, delete and edit modeling and language elements such as entities or locations. It is similar to a spread sheet editor in that it provides maximum visibility of element lists while still allowing each field of a particular element to be directly edited.

File Name

A file name is any name used to identify a file. File names may include a path (e.g., C:\REPORTS\DATA) as well as a terminating period with up to three additional characters as an extension (e.g., DATA.TXT). File names are case insensitive.

Font

A font is a collection of characters sharing the same attributes such as height, width, and typeface. A font determines the appearance of text. Select fonts using the font dialog which shows how the font will appear.

Frame

A frame is a graphical border or background for placing text or displaying values. Frames have a type (raised, recessed, transparent), shape, color and size. Frames are optional and you may use them only to enhance the appearance of the graphic layout. If you do not desire a frame, select Transparent as the type.

Gauge

A gauge is a graphic bar that extends and retracts to represent the current contents of a location. You define a gauge by the fill color, the empty color, and the gauge border color. You also specify the fill direction (up, down, left or right).

Integer Number

An integer number is a whole number ranging from -2,147,483,648 to 2,147,483,647. Integer values may not include commas, so enter the number 5,380 as 5380.

Library Graphic

A library graphic is a graphic defined and edited using the Library Graphics Editor. Use library graphics to represent locations, resources or entities, or as part of the layout background with no association to any model entity.

When used to represent a processing location, library graphics may have entity spots defined for them so entities that enter the location will appear graphically on the entity spot.

You may reference a Library graphic multiple times and each reference may have a unique size, horizontal or vertical inversion, rotation, and color.

Main Entity

The original or input entity used in routing logic. You may create new entities from the main entity using the CREATE statement or through multiple routing blocks.

Main Menus

The main menus are listed along the main menu bar at the top of the application window and provide access to all of the dialogs and commands necessary to build, run, and analyze simulation models.

Name Index

During simulation run time, ProModel converts the name of each location, entity, and resource to a list index number for efficient access and flexibility in assigning names to variables, attributes, etc. The index number corresponds to the order in which the elements appear in the edit table list. If five locations were defined, they will be identified by the index numbers 1 through 5 at run time. A name index may be assigned to a variable or attribute by referencing the name itself (e.g., Attr1 = EntityA). You may also test for a name index by referencing the name (e.g., if Var5 = Loc3).

Note

A note is any comment or other information typed by the user that is for information only and disregarded by ProModel.

Numeric Type

A numeric type specifies whether a numeric expression is of type real or integer. If, for example, a variable is of type “real,” then the variable can only represent some real number.

Numeric Value

A numeric value is any real or integer number such as 3.68 or 52.

Parameter

Parameters are variables used in a subroutine which are local to or have scope only within the subroutine. Arguments or numeric expressions passed to a subroutine are assigned to the parameters for use inside the subroutine.

Park Search

A park search defines the sequence where a resource looks for nodes at which to park after completing a task assuming no other tasks are waiting.

Paths

Paths define the course of travel for entities and resources between locations. You may define a path for a specific entity and routing, or a network of paths shared by several resources and entities. Define movement along a path in units of time, or speed and distance.

Path networks consist of nodes connected by path segments. Any node may have multiple input and output segments.

Positioning Spot

An entity spot is simply a graphic position relative to a location or resource and displays any entities occupying the location or resource.

For any given location you may place one or more entity spots on the graphics layout. Entities to enter a location appear on the first available entity spot in the order they are placed. If an entity enters a location and all the entity spots are filled, the entity will appear on the last entity spot.

For resources, place the entity spot where it appears when a resource carries the entity.

Preemption

Preemption is the act of bumping or replacing an activity currently using a location with an activity of a higher preemptive priority. ProModel handles preempted activities differently depending on the location preempted.

For locations, ProModel puts the preempted activity (any current entity or downtime) in a preemption list for that particular location until it can resume its activity at that specific location unit.

Real Number

A real number is a number ranging from 1.7 X 10 -308 to 1.7 X 10 +308. Real values may not include commas, so enter the number 5,380.5 as 5380.5.

Examples: -2.87563, 844.2, 65.0

Reference

A reference is a name entry in an edit field that references a defined model element. If you change the name of the model element, all references to the element automatically change to reflect the change in name.

Region

A region is a rectangular area on the graphic layout that represents a location. Defining a region is useful when you import a layout from a CAD drawing and you want to designate a portion of the layout to represent a particular processing location. A region should have one or more entity spots associated with it in order to be meaningful.

Resource

A resource is a person or item used to perform an operation or activity. Common resources include human operators, inspectors, forklifts, and other vehicles. Use resources used to perform operations on entities at a location, transport entities between locations, or perform activities on a location during a downtime.

Resource Point

A resource point is a screen position where a resource will appear when it arrives to park or perform a task at a particular node. When a resource arrives at a node, it will appear on that node unless you define a resource point for that resource at that node. Resource points provide a way to have several resources positioned at the same node without all appearing on top of each other.

Routing Priority

A routing priority is the priority given to a routing for accessing a destination when capacity becomes available. ProModel defines the routing priority as part of the Destination field in the Routing Edit Table. A routing priority applies where two or more entities wait at a location for a routing destination to become available. The routing priority breaks a tie when deciding which entity has access to the destination location first when it becomes available. (See Selecting Incoming Entities in the Location Rules Dialog.)

Scroll Arrow

You can use the scroll arrows on either end of a scroll bar to move the contents of a window or list box. Clicking once on a scroll arrow moves the contents one line. Holding down a scroll arrow scrolls continuously.

Scroll Bar

A scroll bar is a Windows control for scrolling the contents of a window. Scroll bars operate in three different ways:

1. Click on the scroll arrows at either end of the scroll bar for incremental scrolling.
2. Click on either side of the scroll box for scrolling one window at a time.
3. Drag the scroll box with the mouse to scroll to a specific position.

Scroll Box

A scroll box is a small box in a scroll bar that shows the position of what is currently in the window or list box relative to the contents of the entire window.

Shell

The temporary entity (representing grouped entities) to which ProModel assigns all costs and statistics until you ungroup the entities. Once ungrouped, ProModel divides all costs among the entities.

Status Lights

A status light is a circle that lights up with different colors depending on the status of the location. You can place a status light anywhere relative to a location for showing the status or current state of the location. At run time, you can display a window showing what status each color represents.

String

A string is a series of characters enclosed in double quotes (e.g., “Station A”). You may use strings to write text to a file or to display a message on the screen.

System Menu

The system menu appears as a dash mark at the upper left corner of most windows. The main function of the system menu is to close the window. To do this, by either double-click on the menu button or click on the button once and select Close from the menu options that appear. ALT + <space bar> also pulls up the system menu.

Text

Text refers to words that you wish to display on the graphic layout. Each specification of text has an associated font, color, frame, and orientation (up, down, left or right). You can rotate only true-type fonts.

Time Statistics

Statistics collected on a time basis (e.g., time in system, average minutes per entry, and average contents). You *cannot* control how ProModel collects these statistics since ProModel bases them solely on the system clock.

Work Search

A work search defines the sequence in which a resource looks for work at locations where work may wait to be performed.

Bibliography

Carson, J. S. “Convincing Users of Model's Validity is Challenging Aspect of Modeler's Job,” Industrial Engineering, June 1986, p. 77.

Conway, Richard, William L. Maxwell, and Steven L. Worona, User's guide to XCELL Factory Modeling System, The Scientific Press, 1986, pp 65-66.

Gordon, Geoffrey, System Simulation, 2nd ed., Prentice-Hall, 1978.

Harrell, Charles; Ghosh, Biman; Bowden, Royce. 2003. *Simulation Using ProModel*. 2nd Edition; McGraw-Hill, Inc.

Harrell, Charles R. and Kerim Tumay, Simulation Made Easy, Industrial Engineering Press, 1995.

Hoover, Stewart V. and Ronald F. Perry, Simulation: A Problem Solving Approach, Addison-Wesley, Reading Massachusetts, 1990.

Knepell, Peter L. and Deborah C. Arangno, Simulation Validation, IEEE Computer Society Press, 1993.

Law, Averill M. and David W. Kelton, Simulation Modeling and Analysis, McGraw-Hill, 1991.

Law, Averill M. “Designing and Analyzing Simulation Experiments,” Industrial Engineering, March 1991, pp. 20-23

Neelamkavil, Francis Computer Simulation and Modeling, John Wiley & Sons, 1987.

Pritsker, Alan B. and Claude Dennis Pegden, Introduction to Simulation and SLAM, John Wiley & Sons, 1979.

Schlesinger, S. “Terminology for Model Credibility,” Simulation, 32(3), 1979, pp.103-104.

Shannon, Robert E., Systems Simulation: The Art and Science, Prentice-Hall, 1975.

Thesen, Arne and Laurel E. Travis, Simulation For Decision Making, West Publishing Company, 1992.

Tumay, Kerim, Business Process Reengineering Using Simulation, Autofact Workshop, 1993.

Index