THE POWER TO KNOW.

SAS/OR® 14.3 User's Guide
Project Management

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2017. *SAS/OR® 14.3 User's Guide: Project Management*. Cary, NC: SAS Institute Inc.

SAS/OR® 14.3 User's Guide: Project Management

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

September 2017

Contents

Credits and Acknowledgments

Credits

Documentation

Writing	Gehan A. Corea, Charles B. Kelly, Radhika V. Kulkarni, Michelle Opp, Lindsey Puryear, Tien-yi D. Shaw, Keqi Yan
Editing	Anne Baxter, Virginia Clark, Ed Huddleston, Donna Sawyer
Documentation Support	Tim Arnold, Natalie Baerlocher, Remya Chandran, Melanie Gratton, Richard Liu, Jianzhe Luo, Michelle Opp, Girish Ramachandra, Daniel Underwood
Technical Review	Marianne Bohinski, Tonya Chapman, Marc-david Cohen, Donna Fulenwider, Phil Gibbs, Barbara J. Hoopes, Tao Huang, Edward P. Hughes, John Jasperse, Charles B. Kelly, Emily Lada, Michelle Opp, Bengt Pederson, Rob Pratt

Software

The procedures in SAS/OR software were implemented by the Operations Research and Development Department. Substantial support was given to the project by other members of the Analytical Solutions Division. Core Development Division, Display Products Division, Graphics Division, and the Host Systems Division also contributed to this product.

In the following list, the names of the developers currently supporting the procedure are listed first. Other developers previously worked on the procedure.

CPM	Lindsey Puryear, Radhika V. Kulkarni
DTREE	Tien-Yi D. Shaw
GANTT	Gehan A. Corea, Lindsey Puryear, Radhika V. Kulkarni
NETDRAW	Lindsey Puryear, Radhika V. Kulkarni
PM	Gehan A. Corea, Radhika V. Kulkarni
EVM Macros	Lindsey Puryear
Microsoft Conversion Macros	Liping Cai, Gehan A. Corea, Keqi Yan

Support Groups

Software Testing	Nitin Agarwal, Marianne Bohinski, Liping Cai, Tao Huang, Edward P. Hughes, John Jasperse, Charles B. Kelly, Emily Lada, Bengt Pederson, Rob Pratt, Tugrul Sanli, Kaihong Xu
Technical Support	Tonya Chapman

Acknowledgments

Many people have been instrumental in the development of SAS/OR software. The individuals acknowledged here have been especially helpful.

G. Jack Theurer	G. Theurer Associates
Lance Broad	New Zealand Ministry of Forestry
Richard Brockmeier	Union Electric Company
Ken Cruthers	Goodyear Tire & Rubber Company
Patricia Duffy	Auburn University

Richard A. Ehrhardt	University of North Carolina at Greensboro
Paul Hardy	Babcock & Wilcox
Don Henderson	ORI Consulting Group
Dave Jennings	Lockheed Martin
Vidyadhar G. Kulkarni	University of North Carolina at Chapel Hill
Wayne Maruska	Basin Electric Power Cooperative
Bruce Reed	Auburn University
Charles Rissmiller	Lockheed Martin
David Rubin	University of North Carolina at Chapel Hill
John Stone	North Carolina State University
Keith R. Weiss	ICI Americas Inc.

The final responsibility for the SAS System lies with SAS Institute alone. We hope that you will always let us know your opinions about the SAS System and its documentation. It is through your participation that SAS software is continuously improved.

Chapter 1

What's New in SAS/OR 14.3

Contents

Overview

SAS/OR 14.3 includes performance improvements in the LP, MILP, QP, and NLP solvers. If you also license SAS Optimization 8.1 (or a later release), you can invoke it and SAS/OR 14.3 from the same SAS session.

SAS Simulation Studio 14.3, a component of SAS/OR 14.3 for Windows environments, adds Linux support (on an experimental basis) and also adds an alternative user interface design.

Mathematical Optimization Updates

Solver Performance Improvements

Several optimization solvers have been updated in SAS/OR 14.3 and improve their performance. The LP, MILP, QP, and NLP solver algorithms all reduce the time they require to solve benchmark optimization problems. These improvements also include the decomposition (DECOMP) algorithm for LP and MILP. You should expect to be able to solve individual optimization problems of these types more quickly. If you invoke one of these optimization solvers repeatedly (for example, within a loop), the reduction in overall solution time should be even more pronounced.

SAS Optimization 8.1, which was introduced in March 2017 as part of SAS Viya 3.2, adds several distributed optimization capabilities and works with many of the same optimization procedures as SAS/OR 14.3. For more information, see the SAS Optimization 8.1 documentation. If you license this product along with SAS/OR 14.3, you can access the capabilities of both products from the same SAS session.

Discrete-Event Simulation Updates

SAS Simulation Studio 14.3, which provides a graphical environment for building and working with discrete-event simulation models, makes two changes.

In addition to its long-established support on Windows, SAS Simulation Studio 14.3 adds Linux support on an experimental basis. In addition, the original user interface design is now accompanied by an optional alternative design. The original design is still the default, but you can select the new design by using the Configuration dialog box. The original design is named Metal, and the new alternative design is named Nimbus.

Chapter 2
Using This Book

Contents

Purpose

SAS/OR User's Guide: Project Management provides a complete reference for the project management procedures in SAS/OR software. This book serves as the primary documentation for the CPM, DTREE, GANTT, NETDRAW, and PM procedures.

This chapter describes the organization of this book and the conventions that are used in the text and example code. To gain full benefit from using this book, familiarize yourself with the information presented in this section and refer to it when needed. The section "Additional Documentation for SAS/OR Software" on page 7 refers to other documents that contain related information.

Organization

Chapter 3 contains a brief overview of the project management procedures in SAS/OR software and provides an introduction to project management methodology and the use of the project management tools in the SAS System. That chapter also describes the flow of data between the procedures and how the components of the SAS System fit together.

The remaining chapters describe the CPM, DTREE, GANTT, NETDRAW, and PM procedures, the earned value management macros, and the Microsoft Project conversion macros. Each procedure description is self-contained; you need to be familiar with only the basic features of the SAS System and with SAS terminology to use most procedures. The statements and syntax necessary to run each procedure are presented in a uniform format throughout this book.

The following list summarizes the types of information provided for each procedure:

Overview provides a general description of what the procedure does. It outlines major capabilities of the procedure and lists all input and output data sets that are used with it.

Getting Started illustrates simple uses of the procedure in a few short examples. It provides introductory hands-on information for the procedure.

Syntax constitutes the major reference section for the syntax of the procedure. First, the statement syntax is summarized. Next, functional summary tables list all the statements and options in the procedure, classified by function. In addition, the online version includes a Dictionary of Options, which provides an alphabetical list of all options. Following these tables, the PROC statement is described, and then all other statements are described in alphabetical order. The mode-specific options (line-printer, full-screen, and graphics options) for the DTREE, GANTT, and NETDRAW procedures are described alphabetically under appropriate subheadings.

Details describes the features of the procedure, including algorithmic details and computational methods. This section explains how the various options interact with each other, describes input and output data sets in greater detail (with definitions of the output variables) and explains the format of printed output, if any.

Examples consists of examples that are designed to illustrate the use of the procedure. Each example includes a description of the problem and lists the options highlighted that are by the example. The example shows the data and the SAS statements needed and includes the output that is produced. You can duplicate the examples by copying the statements and data and running the SAS program. The SAS Sample Library contains the code that is used to run the examples shown in this book; consult your SAS Software representative for specific information about the Sample Library. Cross-reference tables in each chapter list all the options and statements illustrated by the different examples in that chapter.

References lists references that are relevant to the chapter.

Typographical Conventions

This book uses various type styles, as explained by the following list:

roman	is the standard type style used for most text.
UPPERCASE ROMAN	is used for SAS statements, options, and other SAS language elements when they appear in the text. However, you can enter these elements in your own SAS code in lowercase, uppercase, or a mixture of the two. This style is also used for identifying arguments and values (in the syntax specifications) that are literals (for example, to denote valid keywords for a specific option).
UPPERCASE BOLD	is used in the "Syntax" section to identify SAS keywords, such as the names of procedures, statements, and options.
bold	is used to identify menu items.
VariableName	is used for the names of SAS variables and data sets when they appear in the text.
oblique	is used to indicate an option variable for which you must supply a value (for example, DUPLICATE= *dup* indicates that you must supply a value for *dup*).
italic	is used for terms that are defined in the text, for emphasis, and for publication titles.
`monospace`	is used to show examples of SAS statements. In most cases, this book uses lowercase type for SAS code. You can enter your own SAS code in lowercase, uppercase, or a mixture of the two.

Conventions for Examples

Most of the output shown in this book is produced with the following SAS System options:

```
options linesize=80 pagesize=60 nonumber nodate;
```

In addition, most of the graphics output shown in this book are produced with the following options:

```
goptions hpos=80 vpos=32 ypixels=800;
```

The remaining graphics options used in this book depend on the type of output and on the procedure used to create the output. The general options for half-page portrait, full-page portrait, and full-page landscape output are as follows:

Half-Page Portrait

```
goptions hsize=5.75 in vsize=4.0 in;
```

Full-Page Portrait

```
goptions hsize=5.75 in vsize=8.0 in;
```

Full-Page Landscape

```
goptions hsize=8.0 in vsize=5.75 in
   border rotate=landscape;
```

Additional Graphics Options by Procedure

GANTT Procedure

The following PATTERN statements are used to create the color output from PROC GANTT:

```
pattern1 c=green   v=s;
pattern2 c=green   v=e;
pattern3 c=red     v=s;
pattern4 c=magenta v=e;
pattern5 c=magenta v=s;
pattern6 c=cyan    v=s;
pattern7 c=black   v=e;
pattern8 c=blue    v=s;
pattern9 c=brown   v=s;
```

NETDRAW Procedure

The following GOPTIONS and PATTERN statements are used to create the color output from PROC NETDRAW:

```
goptions cback=ligr;
pattern1 v=e c=green;
pattern2 v=e c=red;
pattern3 v=e c=magenta;
pattern4 v=e c=blue;
pattern5 v=e c=cyan;
```

DTREE Procedure

The following GOPTIONS statement is used to create the color output from PROC DTREE:

```
goptions cback=ligr ctext=black;
```

Accessing the SAS/OR Sample Library

The SAS/OR Sample Library includes many examples that illustrate the use of SAS/OR software, including the examples used in this documentation. To access these sample programs from the SAS windowing environment, select **Help** from the main menu and then select **Getting Started with SAS Software**. On the **Contents** tab, expand the **Learning to Use SAS**, **Sample SAS Programs**, and **SAS/OR** items. Then click **Samples**.

Online Documentation

This documentation is available online with the SAS System. To access SAS/OR documentation from the SAS windowing environment, select **Help** from the main menu and then select **SAS Help and Documentation**. On the **Contents** tab, expand the **SAS Products** and **SAS/OR** items. Then expand the book you want to view. You can search the documentation by using the **Search** tab.

You can also access the documentation by going to `http://support.sas.com/documentation`.

Additional Documentation for SAS/OR Software

In addition to *SAS/OR User's Guide: Project Management,* these other documents can be helpful when you are using SAS/OR software:

SAS/OR User's Guide: Bill of Material Processing
provides documentation for the BOM procedure and all bill of material postprocessing SAS macros. The BOM procedure and SAS macros enable you to generate different reports and to perform several transactions to maintain and update bills of material.

SAS/OR User's Guide: Constraint Programming
provides documentation for the constraint programming procedure in SAS/OR software. This book serves as the primary documentation for the CLP procedure.

SAS/OR User's Guide: Local Search Optimization
provides documentation for the local search optimization procedures in SAS/OR software. This book serves as the primary documentation for the GA procedure, which uses genetic algorithms to solve optimization problems, and the OPTLSO procedure, which performs parallel hybrid derivative-free optimization.

SAS/OR User's Guide: Mathematical Programming
provides documentation for the mathematical programming procedures in SAS/OR software. This book serves as the primary documentation for the OPTLP, OPTMILP, OPTMODEL, and OPTQP procedures, the various solvers called by the OPTMODEL procedure, and the MPS-format SAS data set specification.

SAS/OR User's Guide: Mathematical Programming Examples
supplements the *SAS/OR User's Guide: Mathematical Programming* with additional examples that demonstrate best practices for building and solving linear programming, mixed integer linear programming, and quadratic programming problems. The problem statements are reproduced with permission from the book *Model Building in Mathematical Programming* by H. Paul Williams.

SAS/OR User's Guide: Mathematical Programming Legacy Procedures
provides documentation for the older mathematical programming procedures in SAS/OR software. This book serves as the primary documentation for the INTPOINT, LP, NETFLOW, and NLP procedures. Guidelines are also provided on migrating from these older procedures to the newer OPTMODEL family of procedures.

SAS/OR User's Guide: Network Optimization Algorithms
provides documentation for a set of algorithms that can be used to investigate the characteristics of networks and to solve network-oriented optimization problems. This book also documents PROC OPTNET, which invokes these algorithms and provides network-structured formats for input and output data.

SAS Simulation Studio: User's Guide
provides documentation on using SAS Simulation Studio, a graphical application for creating and working with discrete-event simulation models. This book describes in detail how to build and run simulation models and how to interact with SAS software for analysis and with JMP software for experimental design and analysis.

Chapter 3
Introduction to Project Management

Contents

Overview

This chapter briefly describes how you can use SAS/OR software for managing your projects. This chapter is not meant to define all the concepts of project management; several textbooks on project management explain the basic steps involved in defining, planning, and managing projects (for example, Moder, Phillips, and Davis 1983). Briefly, a *project* is defined as any task comprising a set of smaller tasks that need to be performed, either sequentially or in parallel. Projects can be small and last only a few minutes (for instance, running a set of small computer programs), or they can be mammoth and run for several years (for example, the construction of the Channel Tunnel).

SAS/OR software has four procedures that can be used for planning, controlling, and monitoring projects: the CPM and PM procedures for scheduling project activities subject to precedence, time, and resource constraints; the GANTT procedure for displaying the computed schedule; and the NETDRAW procedure for displaying the activity network. These procedures integrate with the SAS System so that you can easily develop a customized project management system.

This chapter gives a brief introduction to the CPM, GANTT, NETDRAW, and PM procedures and shows how you can use the SAS System for project management.

In addition to these four procedures, which are the major tools for the *traditional* functions associated with project management, SAS/OR software also contains a procedure for decision analysis, the DTREE procedure. *Decision analysis* is a tool that attempts to provide an analytic basis for management decisions under uncertainty. It can be used effectively as an integral part of project management methods. A brief introduction to PROC DTREE is provided in the section “Decision Analysis” on page 16.

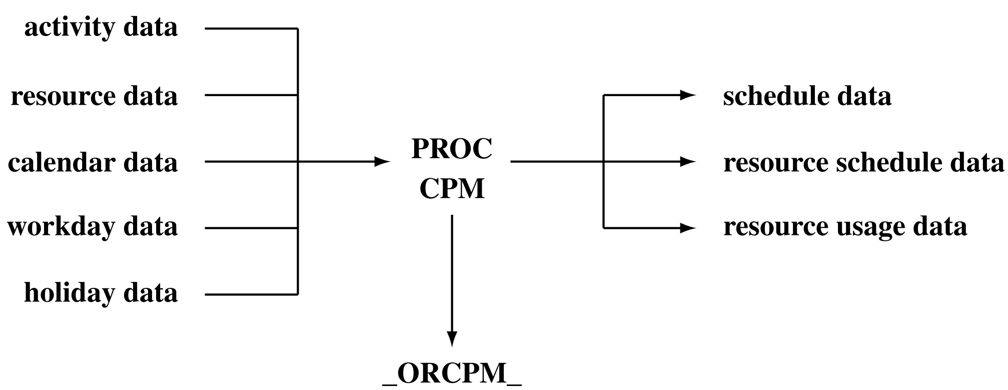
Data Flow

This section provides an overview of how project information is stored in the SAS System in data sets and how these data sets are used by the CPM, GANTT, NETDRAW, and PM procedures. Maintaining the project information in SAS data sets enables you to merge project information easily from several sources, summarize information, subset project data, and perform a wide variety of other operations using any of the many procedures in SAS software. Each of the SAS/OR procedures also defines a SAS macro variable that contains a character string indicating whether or not the procedure terminated successfully. This information is useful when the procedure is one of the steps in a larger program.

The CPM Procedure

PROC CPM does the project scheduling and forms the core of the project management functionality in SAS/OR software. It uses activity precedence, time, and resource constraints, and holiday and calendar information to determine a feasible schedule for the project. The precedence constraints between the activities are described using a network representation, either in Activity-On-Arc (AOA) or Activity-On-Node (AON) notation, and input to PROC CPM in an Activity data set. The two different representations are described in Chapter 4, “The CPM Procedure.” The Activity data set can also specify time constraints on the activities and resource requirement information. The Activity data set is required. Resource availability information can be specified using another data set, referred to here as the Resource data set. Holiday, workday, and other calendar information is contained in the Holiday, Workday, and Calendar data sets; each of these data sets is described in detail in Chapter 4, “The CPM Procedure.” The schedule calculated by PROC CPM using all the input information and any special scheduling options is saved in an output data set, referred to as the Schedule data set. For projects that use resources, individual resource schedules for each activity can be saved in a Resource Schedule output data set. Resource usage information can also be saved in another output data set, referred to as the Usage data set. Figure 3.1 illustrates all the input and output data sets that are possible with PROC CPM. In the same figure, _ORCPM_ is the SAS macro variable defined by PROC CPM.

Figure 3.1 Input and Output Data Flow in PROC CPM



The three output data sets produced by PROC CPM contain all the information about the schedule and the resource usage; these data sets can be used as input to either PROC GANTT or PROC NETDRAW or to any of the several reporting, charting, or plotting procedures in the SAS System.

The Schedule data set can also contain additional project information such as project ID, department and phase information, accounting categories, and so on, in the form of ID variables passed to it from the Activity input data set with the ID statement. These variables can be used to produce customized reports by reordering, subsetting, summarizing, or condensing the information in the Schedule data set in various ways.

The GANTT Procedure

PROC GANTT draws, in line-printer, high-resolution graphics, or full-screen mode, a bar chart of the schedules computed by PROC CPM. Such a bar chart is referred to as a Gantt chart in project management terminology. In addition to the Schedule data set, PROC GANTT can also use the Calendar, Workday, and Holiday data sets (that were used by PROC CPM when scheduling the activities in the project) to mark holidays and weekends and other nonwork periods appropriately on the Gantt chart. You can indicate target dates, deadlines, and other important dates on a Gantt chart by adding CHART variables to the Schedule data set. Furthermore, the GANTT procedure can indicate milestones on the chart by using a DURATION variable in the Schedule data set.

Precedence information can be used by PROC GANTT in either Activity-on-Node or Activity-on-Arc format to produce a Logic bar chart that shows the precedence relationships between the activities. The precedence information, required for drawing the network logic, can be conveyed to PROC GANTT using the Activity data set or a Logic data set, as described in Chapter 8, “The GANTT Procedure.”

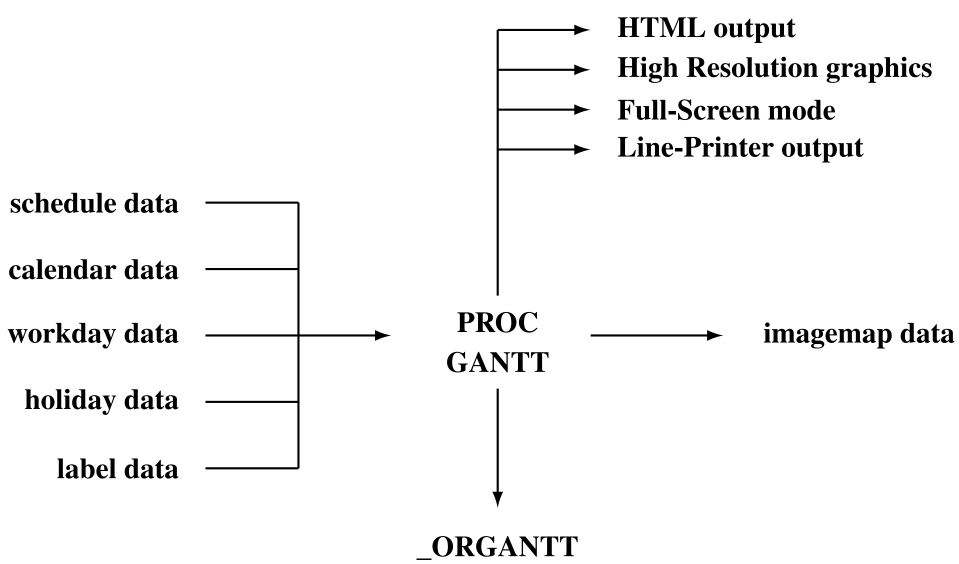
The Gantt procedure also supports an automatic text annotation facility, using the Label data set, which is designed specifically for labeling Gantt charts independently of the SAS/GRAPH Annotate facility. The specifications in this data set enable you to print label strings with a minimum of effort and data entry while providing the capability for more complex chart labeling situations.

The Gantt procedure is Web-enabled. The HTML= option enables you to specify a variable in the Schedule data set that defines a URL for each activity. If you route the Gantt chart to an HTML file using the Output Delivery System, then you can click on a schedule bar and browse text or other descriptive information about the associated activity. You also use this information to create custom HTML files with drill-down graphs.

PROC GANTT also produces an Imagemap data set that contains the outline coordinates for the schedule bars used in the Gantt chart that can be used to generate HTML MAP tags.

As with PROC CPM, PROC GANTT also defines a macro variable named _ORGANTT that has a character string indicating if the procedure terminated successfully. Figure 3.2 illustrates the flow of data in and out of PROC GANTT.

Figure 3.2 Input and Output Data Flow in PROC GANTT



The NETDRAW Procedure

PROC NETDRAW draws project networks. The procedure automatically places the nodes in the network and draws the arcs connecting them, using the (activity, successor) relationship as specified by the Network data set described in Chapter 9, "The NETDRAW Procedure."

The Network data set, used as input to PROC NETDRAW, can be an Activity data set, a Schedule data set, or a Layout data set, as described in Chapter 9.

If a Schedule data set, output by PROC CPM, is used as the Network data set, the network diagram also contains all the schedule times calculated by PROC CPM. The procedure can draw the diagram in line-printer mode as well as in high-resolution graphics mode. Further, you can invoke the procedure in full-screen mode, which enables you to scroll around the network to view different parts of it; in this mode, you can also modify the layout of the network by moving the nodes of the network.

By default, PROC NETDRAW uses the topological ordering of the activity network to determine the X coordinates of the nodes. In a time-based network diagram, the nodes can be ordered according to any SAS date, time, or datetime variable in the Network data set. In fact, PROC NETDRAW enables you to align the nodes according to any numeric variable in this data set, not just the start and finish times.

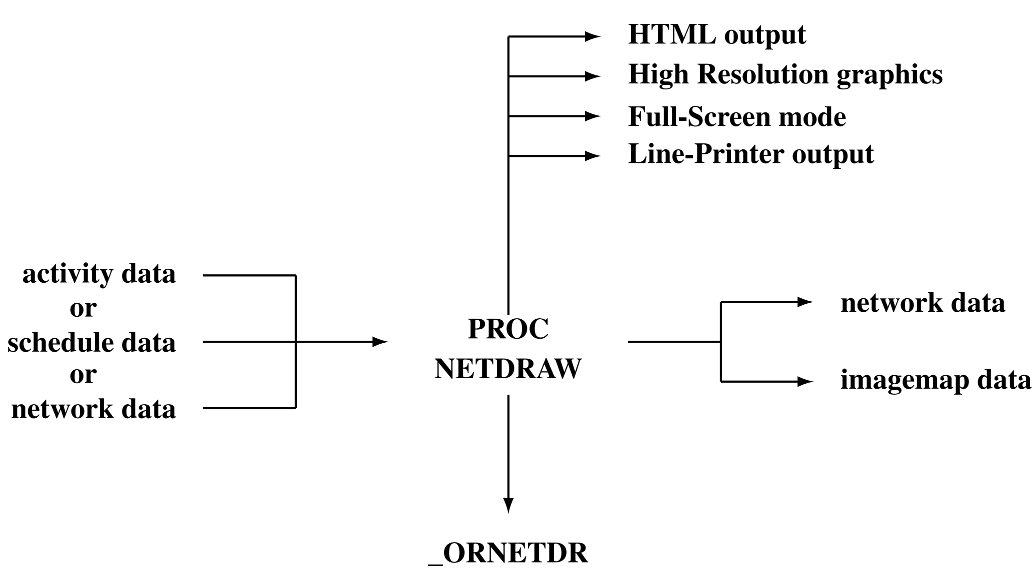
You can produce a zoned network diagram by identifying a ZONE variable in the input data set, which divides the network into horizontal bands or zones. This is useful in grouping the activities of the project according to some appropriate classification. The NETDRAW procedure also draws tree diagrams. This feature can be used to draw work breakdown structures or other organizational diagrams (see Example 3.2).

PROC NETDRAW produces an output data set (Layout data set), which contains the positions of the nodes and the arcs connecting them. This output data set can also be used as an input data set to PROC NETDRAW; this feature is useful when the same project network is drawn several times during the course of a project. You may want to see the updated information drawn on the network every week; you can save computer resources by using the same node placement and arc routing, without having the procedure recompute it every time. PROC NETDRAW defines the macro variable _ORNETDR, which contains a character string indicating if the procedure terminated successfully.

The NETDRAW procedure is also Web-enabled (like PROC GANTT), and it supports the HTML= and IMAGEMAP= options.

Figure 3.3 illustrates the flow of data in and out of PROC NETDRAW.

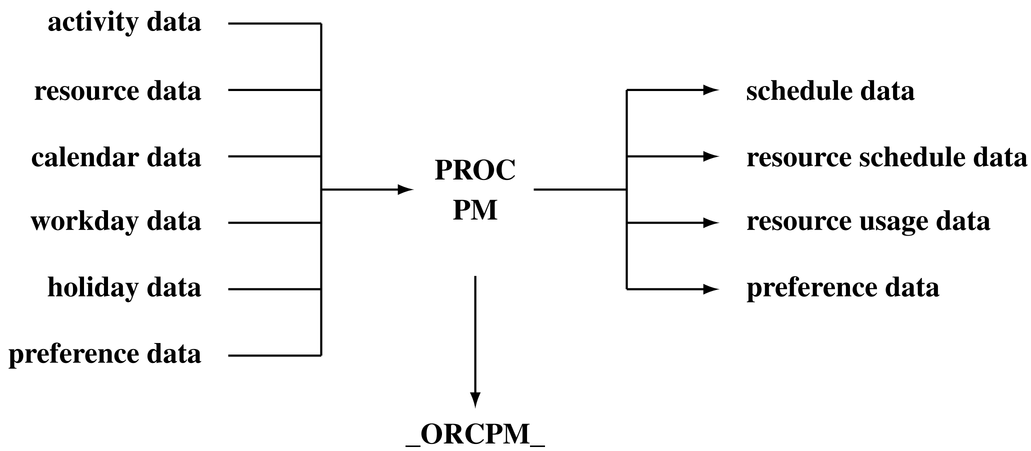
Figure 3.3 Input and Output Data Flow in PROC NETDRAW



The PM Procedure

PROC PM is an interactive procedure that can be used for planning, controlling, and monitoring a project. The syntax and the scheduling features of PROC PM are virtually the same as those of PROC CPM; there are a few differences, which are described in Chapter 5, "The PM Procedure." As far as the flow of data is concerned (see Figure 3.4), the PM procedure supports an additional data set that can be used to save and restore preferences that control the project view. The scheduling engine used by the PM procedure is the same as the one used by PROC CPM; the same macro variable, _ORCPM_, is used to indicate if the schedule was computed successfully.

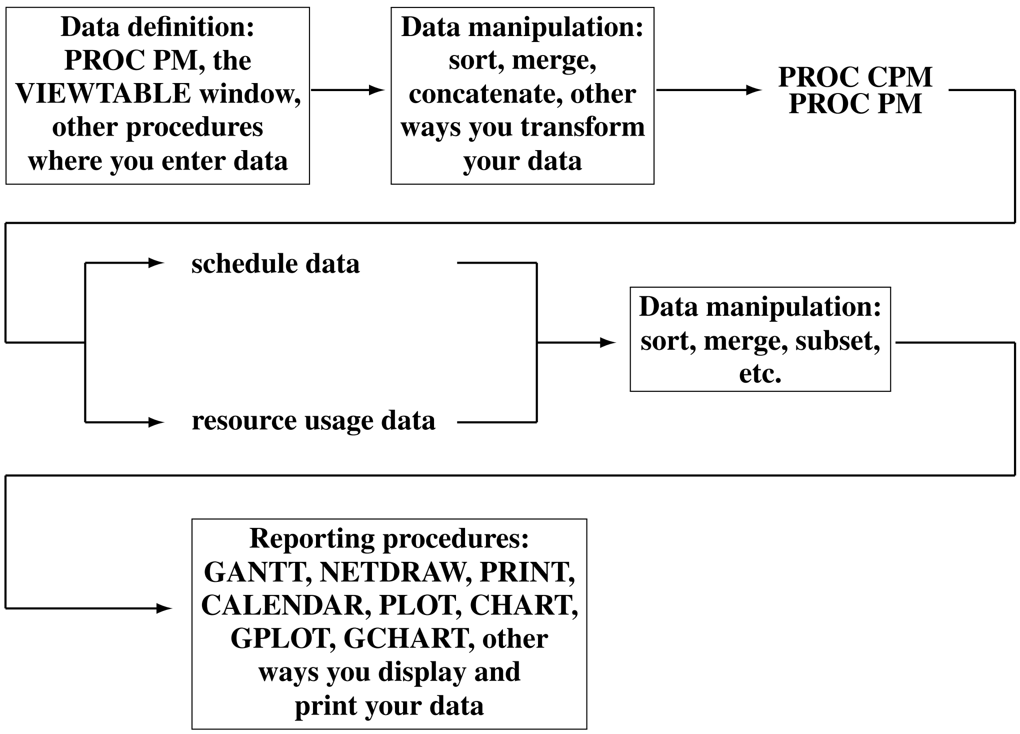
Figure 3.4 Input and Output Data Flow in PROC PM



Communication between Procedures

Figure 3.1, Figure 3.2, Figure 3.3, and Figure 3.4 illustrate the data flow going in and out of each of the four procedures: CPM, GANTT, NETDRAW, and PM, respectively. The data sets described in the previous sections store project information and can be used to communicate project data between the procedures in the SAS System. Figure 3.5 shows a typical sequence of steps in a project management system built around these procedures.

Figure 3.5 Using the SAS System for Project Management

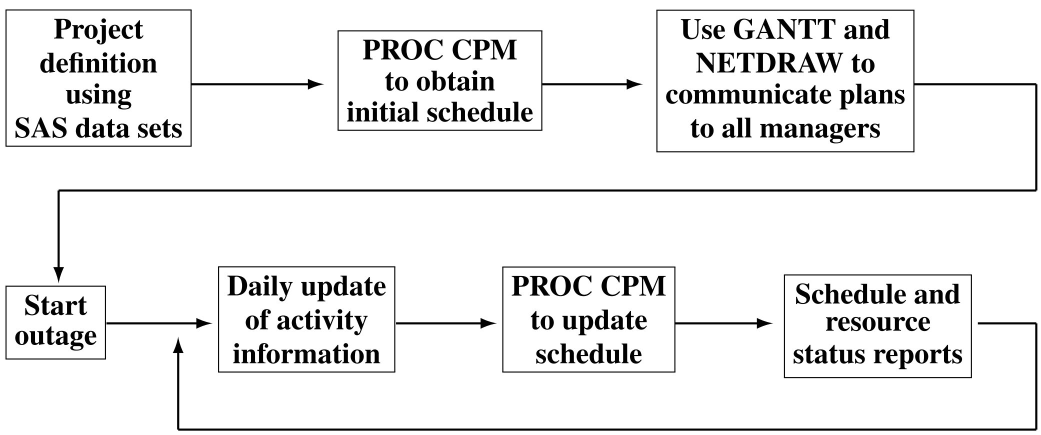


Of course, this is only one possible scenario of the use of these procedures. In addition, you may want to use PROC NETDRAW to check the logic of the network diagram before scheduling the project using PROC CPM. Further, the data flow shown in Figure 3.5 may represent only the first iteration in a continuous scheme for monitoring the progress of a project. As the project progresses, you may update the data sets, including actual start and finish times for some of the activities, invoke PROC CPM again, produce updated Gantt charts and network diagrams, and thus continue monitoring the project.

For example, a project management system designed for scheduling and tracking a major outage at a power plant may include the steps illustrated in Figure 3.6. In the sequences of steps illustrated in both these figures, you can use PROC PM to update most of the activity information using the procedure's graphical user interface.

Thus, SAS/OR software provides four different procedures designed for performing many of the traditional project management tasks; these procedures can be combined in a variety of ways to build a customized comprehensive project management system. The section "Examples" on page 17 illustrates several applications of the procedures in typical project management situations.

Figure 3.6 Scheduling a Power Plant Outage



Decision Support Systems

In addition to the CPM, GANTT, NETDRAW, and PM procedures, which are the major tools for the traditional functions associated with project management, SAS/OR software has several procedures that can be used to create a many-faceted Decision Support System. Traditional CPM/PERT techniques form only one part of effective project management and may be considered as a specialized application of Decision Support Systems (Williams and Boyd 1990). SAS/OR software contains several mathematical programming procedures that can be used to design effective systems for solving inventory control, transportation, network flow, transshipment, product-mix, cutting stock problems, and so on. These procedures are discussed in detail in *SAS/OR User's Guide: Mathematical Programming*.

Decision analysis is another important tool that is receiving recognition as a component of project management. The next section briefly describes PROC DTREE and the role it can play in making important decisions in project management.

Decision Analysis

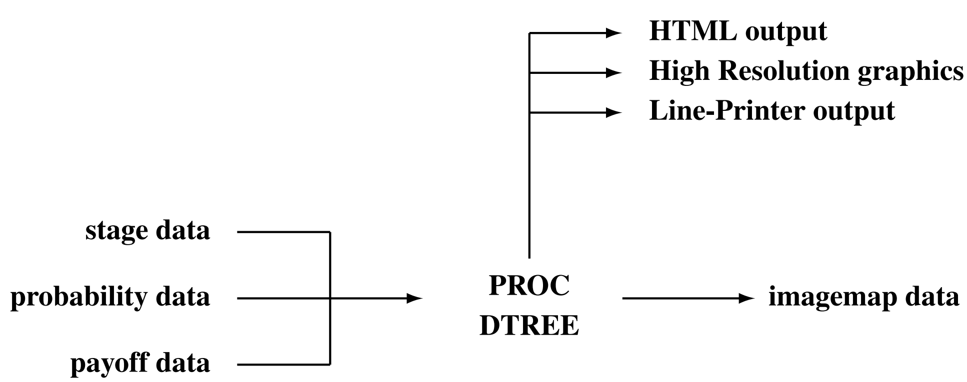
There are several stages in the course of a project when critical decisions are to be made regarding the future path that is to be followed. In fact, the most crucial decision might be to decide at the beginning whether to embark on the project or not. Other important decisions that could benefit from using decision analysis tools may be subcontract awarding, subproject termination in a research and development (R&D) environment, what-if analysis, and so on. Decision analysis techniques can be used effectively in such situations to help make decisions under uncertainty.

The DTREE Procedure

PROC DTREE interprets a decision problem represented in SAS data sets, finds the optimal decisions, and plots on a line printer or a graphics device the decision tree showing the optimal decisions. A decision tree contains two types of nodes: decision nodes and chance nodes. A decision node represents a stage in the problem where a decision is to be made that could lead you along different paths through the tree. A chance node represents a stage in the problem where some uncertain factors result in one of several possible outcomes, once again leading you to different branches of the tree, with associated probabilities.

The structure of a decision model is given in the STAGEIN= data set. This data set, described in detail in Chapter 7, "The DTREE Procedure," specifies the name, type, and attributes of all outcomes for each stage in your model. This is the only data set that is required to produce a diagrammatic representation of your decision problem. To evaluate and analyze your decision model, you need to specify the PROBIN= and PAYOFFS= data sets. The PROBIN= data set specifies the conditional probabilities for every event in your model. The PAYOFFS= data set specifies the value of each possible scenario (sequence of outcomes). The objective is to use the information summarized in these data sets to determine the optimal decision based on some measure of performance. One common objective is to maximize the expected value of the return. Figure 3.7 illustrates the data flow for PROC DTREE.

Figure 3.7 Input and Output Data Flow in PROC DTREE



You can use PROC DTREE to display, evaluate, and summarize your decision problem. The procedure can be used to plot the decision tree in line-printer or graphics mode. The optimal decisions are highlighted on the output. Further, a summary table can be displayed listing all paths through the decision tree along with the cumulative reward and the evaluating values of all alternatives for that path. The summary table indicates the optimal evaluating value for each path with an asterisk. The procedure can also perform sensitivity analysis and what-if analysis. A simple decision problem is described in Example 3.9.

Examples

In this section, a few simple examples illustrate some of the basic data flow concepts described in this chapter. More detailed examples of each procedure are provided in the corresponding chapters and can also be found in *SAS/OR Software: Project Management Examples*.

Example 3.1: Project Definition

Suppose you want to prepare and conduct a market survey (Moder, Phillips, and Davis 1983) to determine the desirability of launching a new product. As a first step, you need to identify the steps involved. Make a list of the tasks that need to be performed and obtain a reasonable estimate of the length of time needed to perform each task. Further, you need to specify the order in which these tasks can be done. The following DATA step creates a SAS data set, survey, representing the project. This Activity data set contains a representation of the Survey project in Activity-On-Node format; a brief discussion of the two types of representations is given in Chapter 4, "The CPM Procedure." The data set contains a variable activity listing the basic activities (tasks) involved, a variable duration specifying the length of time in days needed to perform the tasks, and, for each task, the variables succ1–succ3, which indicate the immediate successors. An ID variable is also included to provide a more informative description of each task. Thus, the activity 'Plan Survey' takes four days. Once the planning is done, the tasks 'Hire Personnel' and 'Design Questionnaire' can begin. The Activity data set also contains a variable named phase associating each activity with a particular phase of the project.

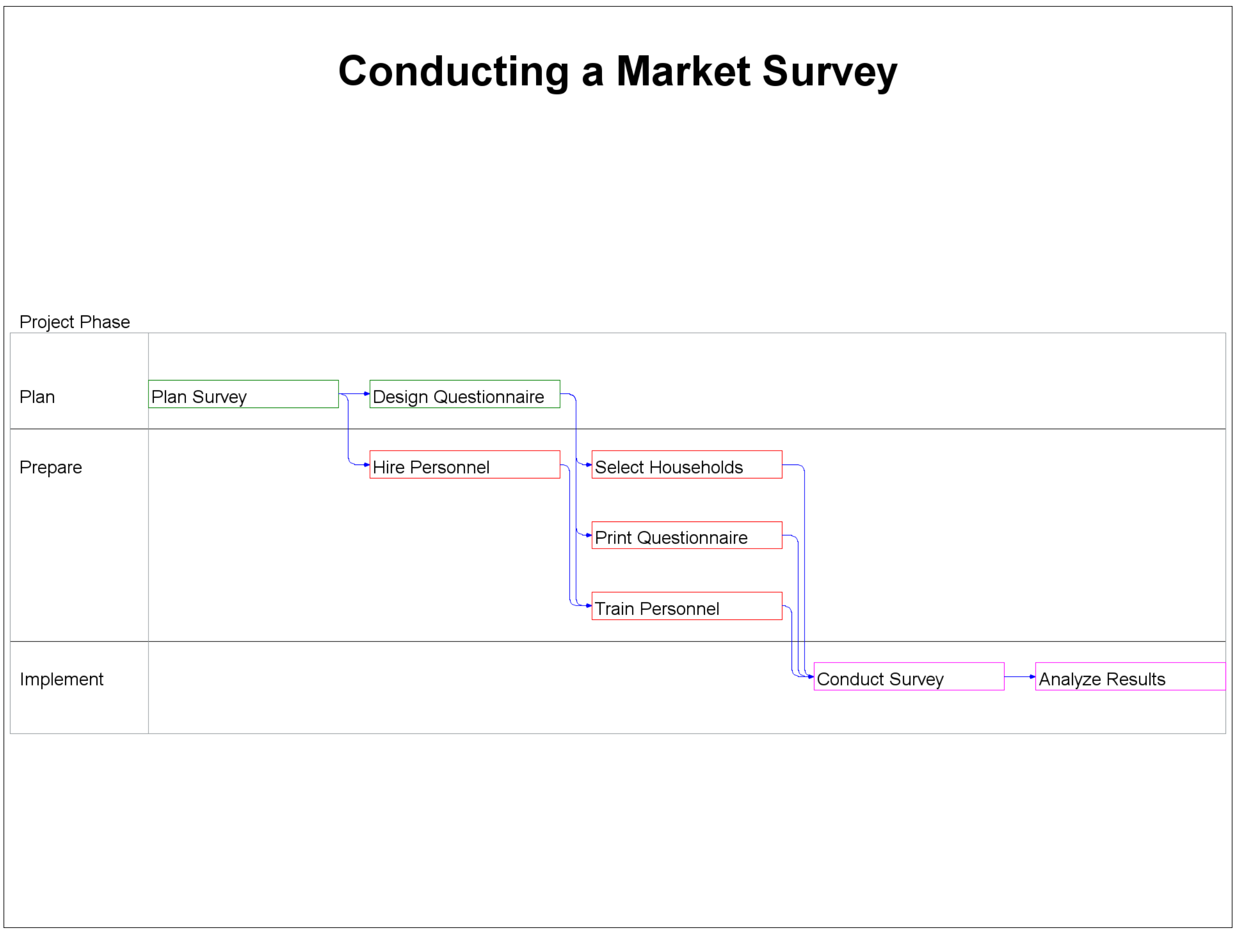
```
data survey;
   input id        $ 1-20
         activity  $ 22-29
         duration
         succ1     $ 34-41
         succ2     $ 43-50
         succ3     $ 52-59
         phase     $ 61-69;
   label phase = 'Project Phase'
         id    = 'Description';
   datalines;
Plan Survey          plan sur 4  hire per design q          Plan
Hire Personnel       hire per 5  trn per                    Prepare
Design Questionnaire design q 3  trn per  select h print q  Plan
Train Personnel      trn per  3  cond sur                   Prepare
Select Households    select h 3  cond sur                   Prepare
Print Questionnaire  print q  4  cond sur                   Prepare
Conduct Survey       cond sur 10 analyze                    Implement
Analyze Results      analyze  6                             Implement
;
```

The data set survey can be input to PROC CPM, which calculates how long the project will take given the current estimates of the durations. As a first step, you may want to graph the project network using PROC NETDRAW. In the initial stages of defining the tasks in a project, it is useful to see how the tasks relate to each other and perhaps modify some of the relationships. The following program invokes PROC NETDRAW; the ZONE= option is used to create a zoned network diagram with the activities grouped according to the phase of the project to which they correspond. The network diagram is shown in Output 3.1.1.

```
goptions hpos=100 vpos=55 border;
pattern1 v=e c=green;
pattern2 v=e c=red;
pattern3 v=e c=magenta;

title ' ';
title2 h=2.5 'Conducting a Market Survey';
```

```
proc netdraw data=survey graphics;
   actnet/act=activity
          succ=(succ1-succ3)
          separatearcs
          xbetween=3 pcompress
          id=(id)
          height=2
          nodefid
          nolabel
          zone=phase
          zonepat
          frame;
   run;
```

Output 3.1.1 Network Diagram of SURVEY Project

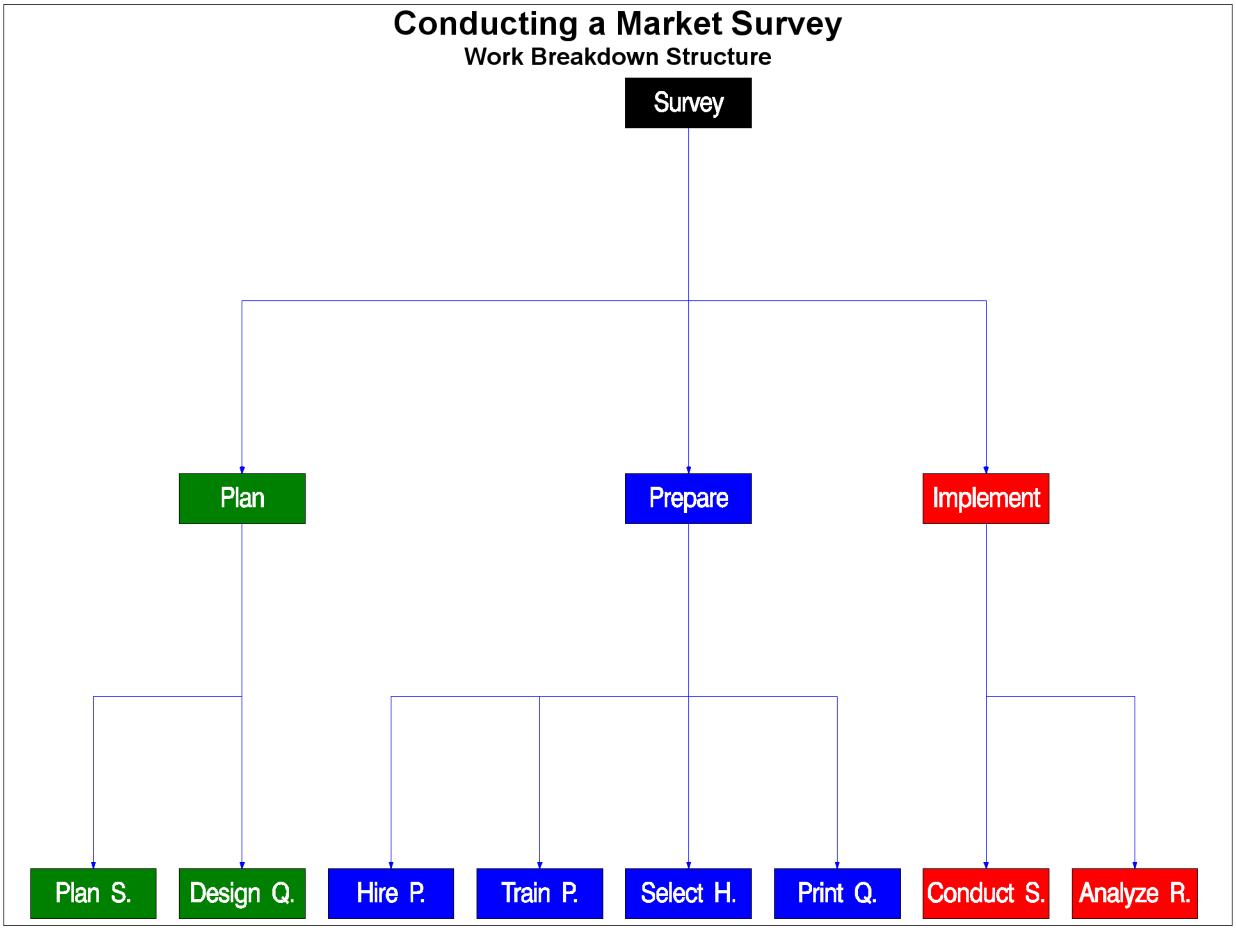


Example 3.2: Work Breakdown Structure

A tree diagram is a useful method of visualizing the work breakdown structure (WBS) of a project. For the survey project, the activities are divided into three phases. In this example, the NETDRAW procedure is used to represent the work breakdown structure of the project. The following program saves the data in a Network data set that is input to PROC NETDRAW. The TREE option is used to draw the WBS structure in the form of a tree (Output 3.2.1).

```
data wbs;
   input parent  $ 1-10
         child   $ 12-21
         style;
   datalines;
Survey     Plan          1
Survey     Prepare       1
Survey     Implement     1
Plan       Plan S.       2
Plan       Design Q.     2
Prepare    Hire P.       3
Prepare    Train P.      3
Prepare    Select H.     3
Prepare    Print Q.      3
Implement  Conduct S.    4
Implement  Analyze R.    4
Plan S.                  2
Design Q.                2
Hire P.                  3
Train P.                 3
Select H.                3
Print Q.                 3
Conduct S.               4
Analyze R.               4
;


pattern1 v=s c=black;
pattern2 v=s c=green;
pattern3 v=s c=blue;
pattern4 v=s c=red;

title  h=2.0   'Conducting a Market Survey';
title2 h=1.4  'Work Breakdown Structure';

proc netdraw data=wbs graphics;
    actnet/act=parent
           succ=child coutline=black tree rotatetext
           ctext=white font=swiss rectilinear
           htext=3 compress rotate
           ybetween=3 xbetween=50 pattern=style
           centerid;
 run;
```

Output 3.2.1 Work Breakdown Structure of SURVEY Project



Example 3.3: Project Scheduling and Reporting

Having defined the project and ensured that all the relationships have been modeled correctly, you can schedule the activities in the project by invoking PROC CPM. Suppose the activities can occur only on weekdays, and there is a holiday on July 4, 2003. Holiday information is passed to PROC CPM using the Holiday data set holidata. The following statements schedule the project to start on July 1, 2003. The early and late start schedules and additional project information are saved in the output data set survschd. The output data set produced by PROC CPM can then be used to generate a variety of reports. In this example, the data set is first sorted by the variable E_START and then displayed using the PRINT procedure (see Output 3.3.1).

```
data holidata;
   format hol date7.;
   hol = '4jul03'd;
run;


proc cpm data=survey date='1jul03'd out=survschd
         interval=weekday holidata=holidata;
   activity   activity;
```

```
   successor  succ1-succ3;
   duration   duration;
   id         id phase;
   holiday    hol;
   run;

proc sort;
   by e_start;
run;

title  'Conducting a Market Survey';
title2 'Early and Late Start Schedule';

proc print;
run;
```

Output 3.3.1 Project Schedule: Listing

Conducting a Market Survey
Early and Late Start Schedule

Obs	activity	succ1	succ2	succ3	duration	id	phase
1	plan sur	hire per	design q		4	Plan Survey	Plan
2	hire per	trn per			5	Hire Personnel	Prepare
3	design q	trn per	select h	print q	3	Design Questionnaire	Plan
4	select h	cond sur			3	Select Households	Prepare
5	print q	cond sur			4	Print Questionnaire	Prepare
6	trn per	cond sur			3	Train Personnel	Prepare
7	cond sur	analyze			10	Conduct Survey	Implement
8	analyze				6	Analyze Results	Implement

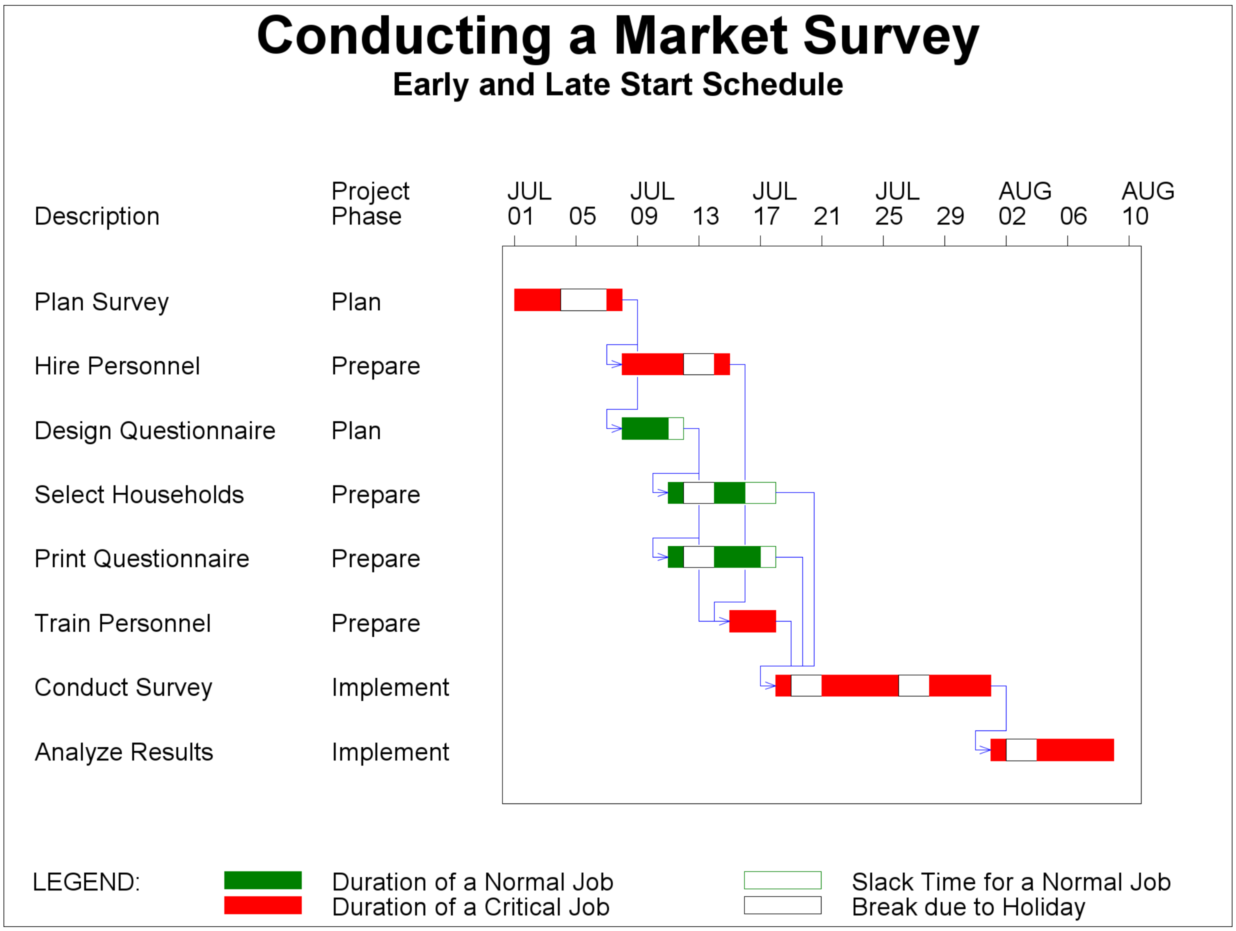
Obs	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	01JUL03	07JUL03	01JUL03	07JUL03	0	0
2	08JUL03	14JUL03	08JUL03	14JUL03	0	0
3	08JUL03	10JUL03	09JUL03	11JUL03	1	0
4	11JUL03	15JUL03	15JUL03	17JUL03	2	2
5	11JUL03	16JUL03	14JUL03	17JUL03	1	1
6	15JUL03	17JUL03	15JUL03	17JUL03	0	0
7	18JUL03	31JUL03	18JUL03	31JUL03	0	0
8	01AUG03	08AUG03	01AUG03	08AUG03	0	0

The schedule produced by PROC CPM is then graphed by invoking PROC GANTT, as shown in the following code. The CALENDAR procedure or NETDRAW procedure can also be used to display the schedule. The Gantt chart produced is shown in Output 3.3.2. Note that the precedence relationships are displayed on the Gantt chart.

```
goptions hpos=80 vpos=43;
pattern1 c=green v=s;     /* duration of a non-critical activity   */
pattern2 c=green v=e;     /* slack time for a noncrit. activity    */
pattern3 c=red   v=s;     /* duration of a critical activity       */
pattern4 c=magenta v=e;   /* slack time for a supercrit. activity  */
pattern5 c=magenta v=s;   /* duration of a supercrit. activity     */
```

```
pattern6 c=cyan v=s;      /* actual duration of an activity       */
pattern7 c=black v=e;     /* break due to a holiday               */

title c=black h=2.5 'Conducting a Market Survey';
title2 c=black h=1.5 'Early and Late Start Schedule';
proc gantt graphics data=survschd holidata=holidata;
   chart / holiday=(hol) interval=weekday
           skip=2 height=1.4 nojobnum
           compress noextrange
           activity=activity succ=(succ1-succ3)
           cprec=blue caxis=black ;
   id   id phase;
   run;
```

Output 3.3.2 Gantt Chart of SURVEY Project



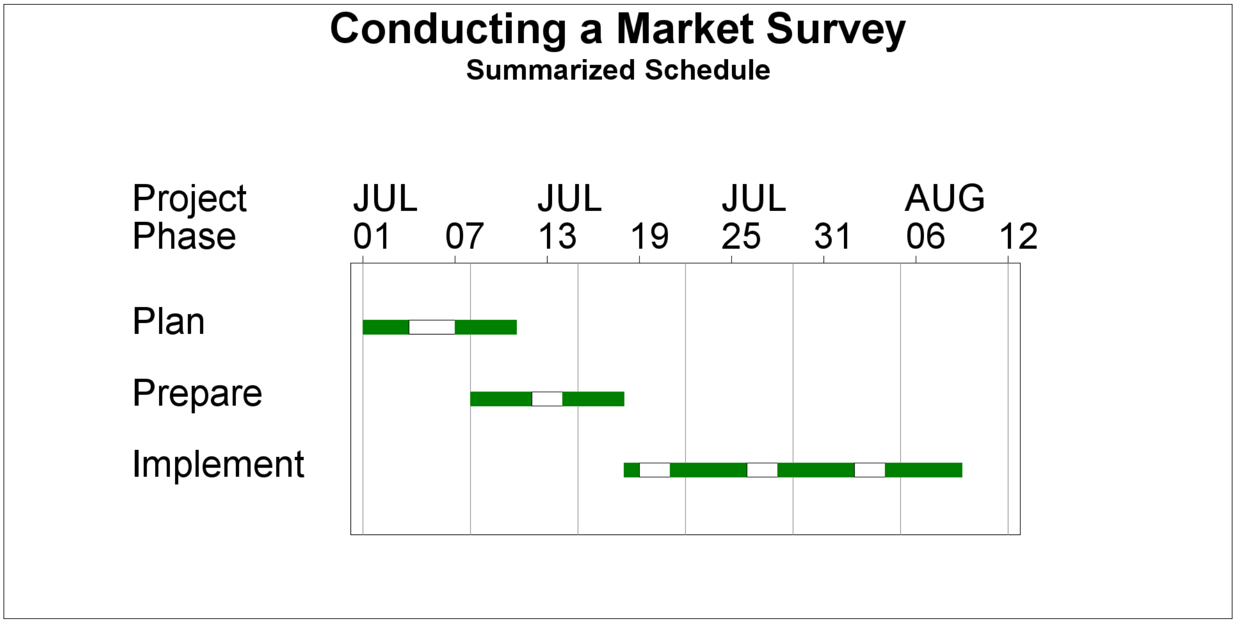
Example 3.4: Summary Report

As mentioned in the section "Data Flow" on page 10, the output data set can be manipulated in several different ways. You can subset the project data to report progress on selected activities, or you can produce reports sorted by a particular field or grouped according to a natural division of the project activities. For large projects, you may want to get a summarized view of the schedule, with the start and finish times of only the major phases of the project.

For the survey project, suppose that you want a condensed report, containing only information about the start and finish times of the three different phases of the project. The following program summarizes the information in the data set survschd and produces a Gantt chart of the summarized schedule (shown in Output 3.4.1).

```
proc sort data=survschd;
   by phase;
   run;

proc summary data=survschd;
   by phase;
   output out=sumsched min(e_start)= max(e_finish)= ;
   var e_start e_finish;
   run;

proc sort data=sumsched;
   by e_start;
   format e_start e_finish date7.;
   run;

pattern1 c=green v=s;
pattern2 c=green v=e;
pattern3 c=red   v=s;
pattern4 c=magenta v=e;
pattern5 c=magenta v=s;
pattern6 c=cyan v=s;
pattern7 c=black v=e;

goptions hpos=80 vpos=43;

title c=black h=3 'Conducting a Market Survey';
title2 c=black h=2 'Summarized Schedule';

proc gantt data=sumsched graphics
     holidata=holidata;
   id phase;
   chart / nojobnum
           nolegend
           interval=weekday
           height=2.0 skip=4
           ref='01jul03'd to '15aug03'd by week
           caxis=black
           cref=gray
           holiday=(hol);
   run;
```

Output 3.4.1 Summary Gantt Chart of SURVEY Project



Example 3.5: Resource-Constrained Scheduling

The previous two examples illustrated some of the reports that can be generated using the Schedule output data set produced by PROC CPM. This section illustrates the use of PROC CPM to perform resource-constrained scheduling and to obtain a resource Usage output data set for generating reports of resource utilization during the course of a project. A primary concern in data processing centers is the number of processors needed to perform various tasks. Given a series of programming tasks, a common question faced by a data center operator is how to allocate computer resources to the various tasks.

Consider a simple job that involves sorting six data sets A, B, C, D, E, and F, merging the first three into one master data set, merging the last three into another comparison data set, and then comparing the two merged data sets. The precedence constraints between the activities (captured by the variables task and succ), the time required by the activities (the variable dur), and the resource required (the variable processor) are shown in the following code:

```
data program;
   format task $8. succ $8. ;
   input task & succ & dur processor;
   datalines;
Sort A     Merge 1     5   1
Sort B     Merge 1     4   1
Sort C     Merge 1     3   1
Sort D     Merge 2     6   1
Sort E     Merge 2     4   1
Sort F     Merge 2     6   1
Merge 1    Compare     5   1
Merge 2    Compare     4   1
Compare    .           5   1
;
```

If the programming project is scheduled (in absolute units) without any resource constraints, it will take 15 time units for completion and will require a maximum availability of six processors. Suppose now that only two processors are available. The resin data set limits the availability of the resource to 2, and PROC CPM is invoked with two input data sets (Activity data set program and Resource data set resin) to produce a resource-constrained schedule.

PROC CPM produces two output data sets. The Schedule data set (progschd) contains the resource-constrained schedule (S_START and S_FINISH variables) in addition to the early and late start unconstrained schedules. The Usage data set (progrout) shows the number of processors required at every unit of time, if the early start schedule or the late start schedule or the resource-constrained schedule were followed, in the variables eprocessor, lprocessor, and rprocessor, respectively; the variable aprocessor shows the number of processors remaining after resource allocation. The two output data sets are displayed in Output 3.5.1.

```
data resin;
   input per processor;
   datalines;
0  2
;

proc cpm data=program resin=resin
         out=progschd resout=progrout;
   activity  task;
   duration  dur;
   successor succ;
   resource  processor/per=per;
   run;

title 'Scheduling Programming Tasks';
title2 'Data Set PROGSCHD';
proc print data=progschd;
   run;

title2 'Data Set PROGROUT';
proc print data=progrout;
   run;
```

The Schedule and Usage data sets, displayed in Output 3.5.1, can be used to generate any type of report concerning the schedules or processor usage. In the following program, the unconstrained and constrained schedules are first plotted using PROC GANTT (see Output 3.5.2).

Output 3.5.1 Data Sets PROGSCHD and PROGROUT

Scheduling Programming Tasks
Data Set PROGSCHD

Obs	task	succ	dur	processor	S_START	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
1	Sort A	Merge 1	5	1	0	5	0	5	0	5
2	Sort B	Merge 1	4	1	6	10	0	4	1	5
3	Sort C	Merge 1	3	1	10	13	0	3	2	5
4	Sort D	Merge 2	6	1	0	6	0	6	0	6
5	Sort E	Merge 2	4	1	11	15	0	4	2	6
6	Sort F	Merge 2	6	1	5	11	0	6	0	6
7	Merge 1	Compare	5	1	13	18	5	10	5	10
8	Merge 2	Compare	4	1	15	19	6	10	6	10
9	Compare		5	1	19	24	10	15	10	15

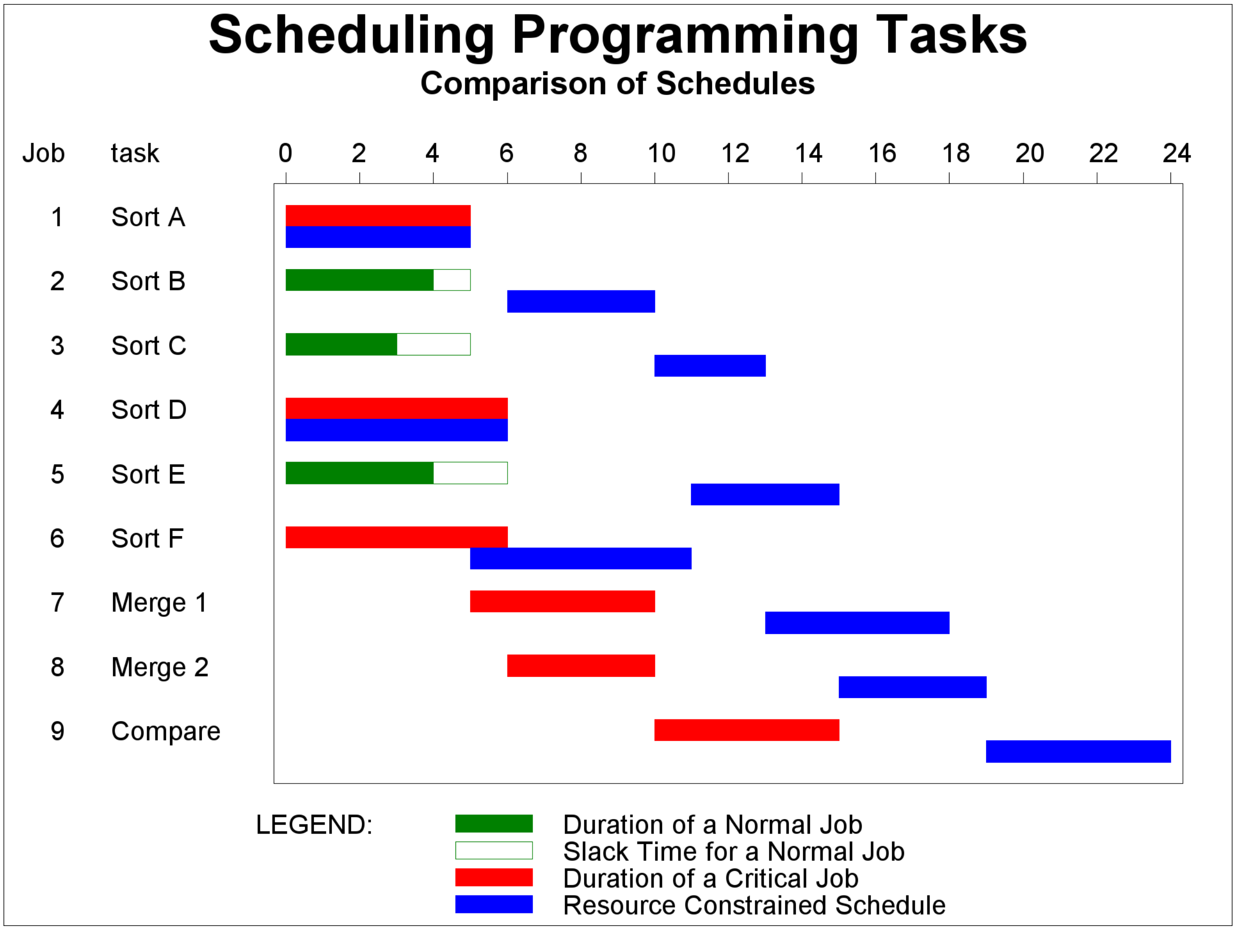
Scheduling Programming Tasks
Data Set PROGROUT

Obs	_TIME_	Eprocessor	Lprocessor	Rprocessor	Aprocessor
1	0	6	3	2	0
2	1	6	4	2	0
3	2	6	6	2	0
4	3	5	6	2	0
5	4	3	6	2	0
6	5	3	4	2	0
7	6	2	2	2	0
8	7	2	2	2	0
9	8	2	2	2	0
10	9	2	2	2	0
11	10	1	1	2	0
12	11	1	1	2	0
13	12	1	1	2	0
14	13	1	1	2	0
15	14	1	1	2	0
16	15	0	0	2	0
17	16	0	0	2	0
18	17	0	0	2	0
19	18	0	0	1	1
20	19	0	0	1	1
21	20	0	0	1	1
22	21	0	0	1	1
23	22	0	0	1	1
24	23	0	0	1	1
25	24	0	0	0	2

```
goptions hpos=80 vpos=43;

title h=2.5 'Scheduling Programming Tasks';
title2 h=1.5 'Comparison of Schedules';
proc gantt data=progschd graphics;
```

```
    chart / height=1.5 increment=2 caxis=black;
    id task;
    run;
```

Output 3.5.2 Gantt Chart Comparing Schedules

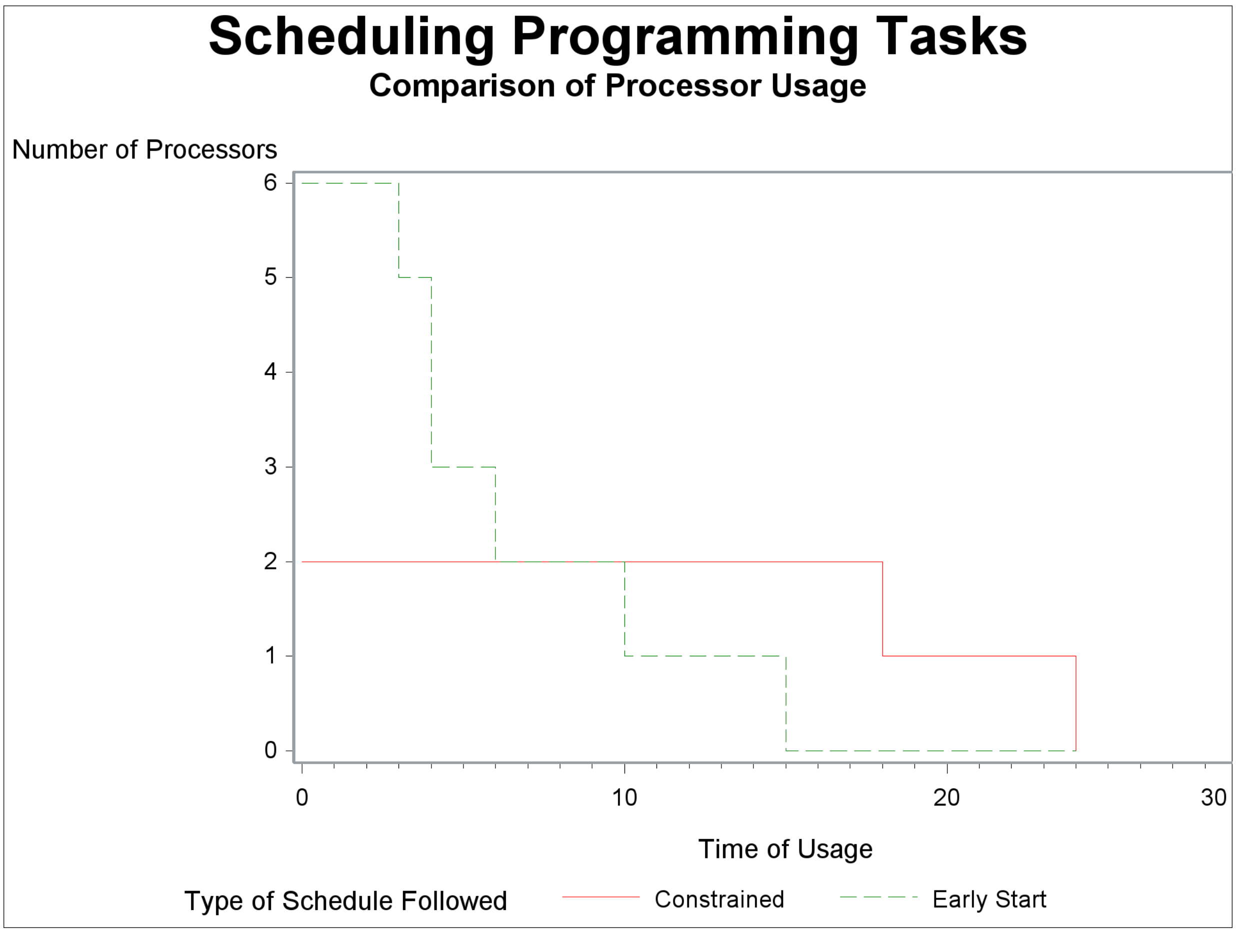


Next, the GPLOT procedure is invoked using the Usage data set to compare the unconstrained and the constrained usage of the resource (see Output 3.5.3).

```
/* Create a data set for use with PROC GPLOT */
data plotout;
    set progrout;
    label _time_='Time of Usage';
    label processor='Number of Processors';
    label resource='Type of Schedule Followed';
    resource='Constrained';
    processor=rprocessor; output;
    resource='Early Start';
    processor=eprocessor; output;
    run;
```

```
axis1 minor=none width=3;
axis2 length=80 pct;
symbol1 i=steplj c=red;
symbol2 i=steplj l=3 c=green;

title2 h=1.5 'Comparison of Processor Usage';
proc gplot data=plotout;
   plot processor * _time_ = resource/ vaxis=axis1
                                       haxis=axis2
                                       caxis=black;
   run;
```

Output 3.5.3 Plot Comparing Resource Usage

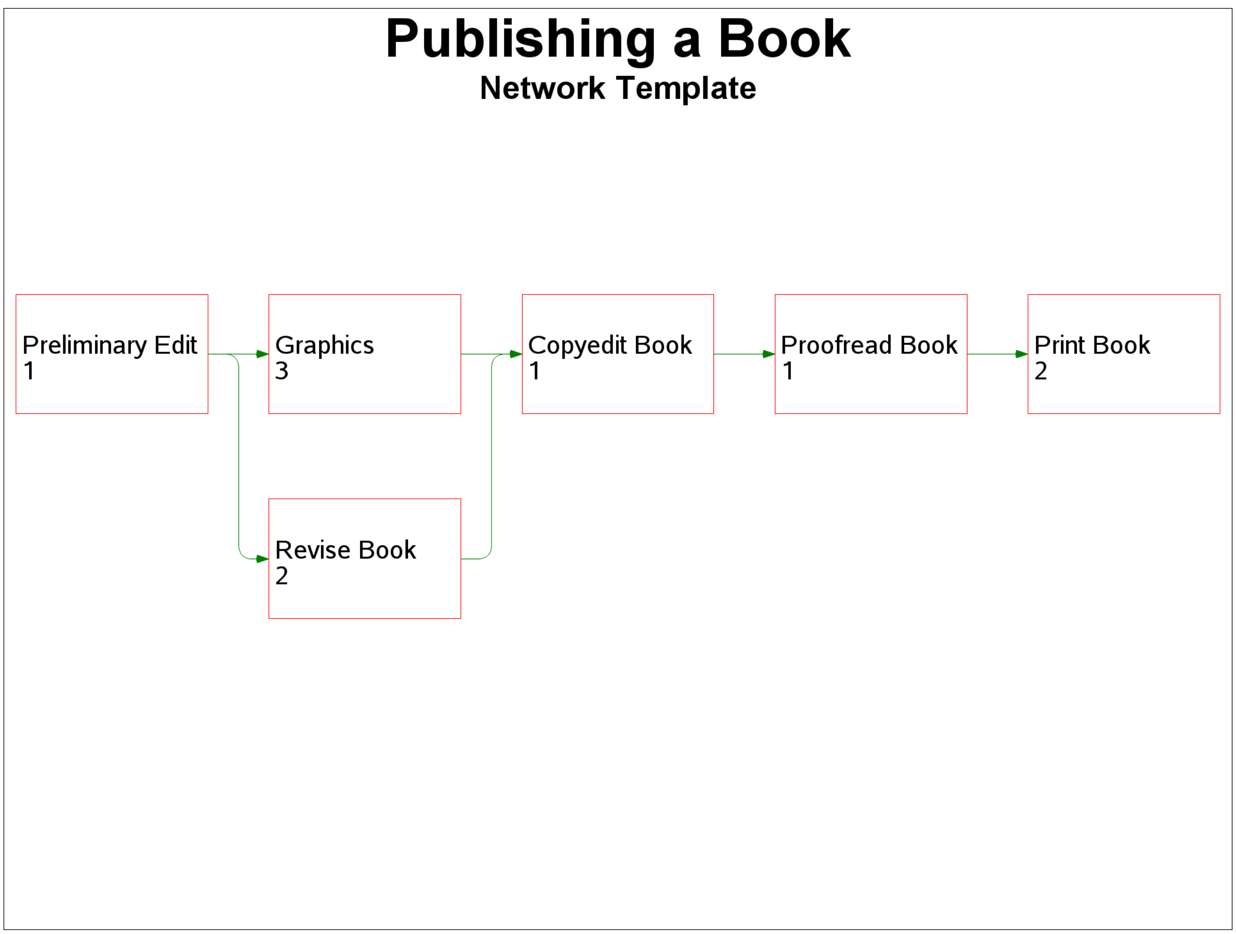


Example 3.6: Multiple Projects

Often a project is divided into several subprojects, each of which is then broken into activities with precedence constraints. For reporting or accounting purposes, it may be essential to group activities or to aggregate the information pertaining to activities in a given group. Sometimes, totally different projects use a common pool of resources and you may want to schedule all the projects using the common pool; you may want to vary the priority with which the resources are allotted to the activities on the basis of the projects to which they belong. Often, you have several projects that are essentially the same, with only a few minor differences; these projects may also share a common pool of resources. In such cases, you may want to have a project *template* listing all the activities and their precedence relationships; for each specific project you can copy the template, make any modifications that are necessary for the given scenario, and determine the project schedule accordingly.

This example illustrates some of these possibilities for a multiproject scenario. The project is first scheduled using PROC CPM, and then the PM procedure is used with the same input data set to illustrate the project displayed in the PM Window.

Output 3.6.1 Network Diagram for Project Book



Consider a publishing company that accepts manuscripts from different authors for publication. The publication of each book can be treated as a project. Thus, at a given point in time, several projects, almost identical in nature, may be in progress. Some of the resources that may be needed are a technical editor, a copyeditor, and a graphic artist. All the books that are currently being worked on share a common pool of these resources. This example uses a simplified version of such a scenario to illustrate some of the ways in which you can handle multiple projects competing for the same pool of resources.

The network in Output 3.6.1 represents some of the tasks required to publish one book and the precedence constraints among these tasks; the durations in the diagram are in weeks. Suppose that the *generic* project data are in the data set book, which is displayed in Output 3.6.2. This data set is used as a *template* for creating the Activity data set for any book publishing project.

Suppose that the company is working on two books simultaneously. The editor and artist must now allocate their time between the two books. The following program uses the *template* data set book to create Activity data sets book1 and book2 corresponding to the publication of each book. Any modifications to the generic project data can be made in the DATA step or by using PROC PM. In this example, the duration for the first activity, ‘Preliminary Edit,’ is changed to two weeks for the second book. The two Activity data sets book1 and book2 are also displayed in Output 3.6.2.

```
data book1;
   length act $6. succ $6.;
   set book;
   subproj = "Book 1";
   act = "B1"||task;
   if succ ^= " " then succ = "B1"||succ;
   run;

data book2;
   length act $6. succ $6.;
   set book;
   subproj = "Book 2";
   act  = "B2"||task;
   if act = "B2PEDT" then dur = 2;
   if succ ^= " "    then succ = "B2"||succ;
   run;

title 'Publishing Book 1';
proc print data=book1;
   var subproj task act succ id dur editor artist;
   run;

title 'Publishing Book 2';
proc print data=book2;
   var subproj task act succ id dur editor artist;
   run;
```

Output 3.6.2 Template and Activity Data Sets for Book Publishing Example

Publishing Book 1

Obs	subproj	task	act	succ	id	dur	editor	artist
1	Book 1	PEDT	B1PEDT	B1REV	Preliminary Edit	1	1	.
2	Book 1	PEDT	B1PEDT	B1GRPH	Preliminary Edit	1	1	.
3	Book 1	REV	B1REV	B1CEDT	Revise Book	2	1	.
4	Book 1	GRPH	B1GRPH	B1CEDT	Graphics	3	.	1
5	Book 1	CEDT	B1CEDT	B1PRF	Copyedit Book	1	1	.
6	Book 1	PRF	B1PRF	B1PRNT	Proofread Book	1	1	.
7	Book 1	PRNT	B1PRNT		Print Book	2	.	.

Publishing Book 2

Obs	subproj	task	act	succ	id	dur	editor	artist
1	Book 2	PEDT	B2PEDT	B2REV	Preliminary Edit	2	1	.
2	Book 2	PEDT	B2PEDT	B2GRPH	Preliminary Edit	2	1	.
3	Book 2	REV	B2REV	B2CEDT	Revise Book	2	1	.
4	Book 2	GRPH	B2GRPH	B2CEDT	Graphics	3	.	1
5	Book 2	CEDT	B2CEDT	B2PRF	Copyedit Book	1	1	.
6	Book 2	PRF	B2PRF	B2PRNT	Proofread Book	1	1	.
7	Book 2	PRNT	B2PRNT		Print Book	2	.	.

As a next step, the data sets for the two subprojects are combined to form an Activity data set for the entire project. A variable priority is assigned the value '1' for activities pertaining to the first book and the value '2' for those pertaining to the second one. In other words, Book 1 has priority over Book 2. The Resource data set specifies the availability for each of the resources to be 1. The input data sets, books and resource, are displayed in Output 3.6.3.

```
data books;
   set book1 book2;
   if subproj = "Book 1" then priority = 1;
   else priority = 2;
   run;

data resource;
   input avdate & date7. editor artist;
   format avdate date7.;
   datalines;
1jan03   1   1
;

title 'Publishing Books 1 and 2';
proc print data=books;
   var subproj priority task act succ id dur editor artist;
   run;

title 'Resources Available';
proc print data=resource;
   run;
```

Output 3.6.3 Input Data Sets for Book Publishing Example

Publishing Books 1 and 2

Obs	subproj	priority	task	act	succ	id	dur	editor	artist
1	Book 1	1	PEDT	B1PEDT	B1REV	Preliminary Edit	1	1	.
2	Book 1	1	PEDT	B1PEDT	B1GRPH	Preliminary Edit	1	1	.
3	Book 1	1	REV	B1REV	B1CEDT	Revise Book	2	1	.
4	Book 1	1	GRPH	B1GRPH	B1CEDT	Graphics	3	.	1
5	Book 1	1	CEDT	B1CEDT	B1PRF	Copyedit Book	1	1	.
6	Book 1	1	PRF	B1PRF	B1PRNT	Proofread Book	1	1	.
7	Book 1	1	PRNT	B1PRNT		Print Book	2	.	.
8	Book 2	2	PEDT	B2PEDT	B2REV	Preliminary Edit	2	1	.
9	Book 2	2	PEDT	B2PEDT	B2GRPH	Preliminary Edit	2	1	.
10	Book 2	2	REV	B2REV	B2CEDT	Revise Book	2	1	.
11	Book 2	2	GRPH	B2GRPH	B2CEDT	Graphics	3	.	1
12	Book 2	2	CEDT	B2CEDT	B2PRF	Copyedit Book	1	1	.
13	Book 2	2	PRF	B2PRF	B2PRNT	Proofread Book	1	1	.
14	Book 2	2	PRNT	B2PRNT		Print Book	2	.	.

Resources Available

Obs	avdate	editor	artist
1	01JAN03	1	1

PROC CPM is then invoked to schedule the project to start on January 1, 2003. The PROJECT statement is used to indicate the subproject to which each activity belongs. The data set bookschd (displayed in Output 3.6.4) contains the schedule for the entire project. The ADDACT option on the PROC CPM statement adds observations for each of the subprojects, 'Book 1' and 'Book 2,' as well as one observation for the entire project. These observations are added at the end of the list of the observations corresponding to the observations in the input data set. The Usage data set booksout is also displayed in Output 3.6.4.

```
proc cpm data=books resin=resource
         out=bookschd resout=booksout
         date='1jan03'd interval=week
         addact;
   act      act;
   dur      dur;
   succ     succ;
   resource editor artist / per=avdate avp rcp
                            rule=actprty actprty=priority
                            delayanalysis;
   id       id task;
   project  subproj;
   run;
```

Compare the E_START and S_START schedules (in the data set bookschd) and note that on January 1, the activity 'B1PEDT' for Book1 is scheduled to start while the preliminary editing of book 2 (activity B2PEDT) has been postponed, due to subproject 'Book 1' having priority over subproject 'Book 2.' On January 22, there is no activity belonging to subproject 'Book 1' that demands an editor; thus, the activity 'B2PEDT' is scheduled to start on that day. As a result, the editor is working on an activity in the second project for two weeks starting from January 22, 2003; when 'B1CEDT' is ready to start, the editor is not available, causing a delay in this activity. Thus, even though the first book has priority over the second book, the scheduling algorithm does not keep a resource waiting for activities in the first project. However, if you enable activity splitting, you can reclaim the resource for the first book by allowing activities in the second project to be split, if necessary. For details regarding the scheduling algorithm allowing splitting of activities, see Chapter 4, "The CPM Procedure."

NOTE: The entire project finishes on April 1, 2003; resource constraints have delayed project completion by four weeks. The variable R_DELAY in the Schedule data set bookschd indicates the amount of delay in weeks caused by resource constraints. The value of R_DELAY does not include any delay in the activity that is caused by a resource delay in one of its predecessors. See Example 4.15 in Chapter 4, "The CPM Procedure," for more details about the R_DELAY variable.

Output 3.6.4 Data Sets BOOKSCHD and BOOKSOUT

Schedule for Project BOOKS

Obs	subproj	PROJ_DUR	PROJ_LEV	act	succ	dur	id	task	editor	artist	S_START
1	Book 1	.	2	B1PEDT	B1REV	1	Preliminary Edit	PEDT	1	.	01JAN03
2	Book 1	.	2	B1PEDT	B1GRPH	1	Preliminary Edit	PEDT	1	.	01JAN03
3	Book 1	.	2	B1REV	B1CEDT	2	Revise Book	REV	1	.	08JAN03
4	Book 1	.	2	B1GRPH	B1CEDT	3	Graphics	GRPH	.	1	08JAN03
5	Book 1	.	2	B1CEDT	B1PRF	1	Copyedit Book	CEDT	1	.	05FEB03
6	Book 1	.	2	B1PRF	B1PRNT	1	Proofread Book	PRF	1	.	12FEB03
7	Book 1	.	2	B1PRNT		2	Print Book	PRNT	.	.	19FEB03
8	Book 2	.	2	B2PEDT	B2REV	2	Preliminary Edit	PEDT	1	.	22JAN03
9	Book 2	.	2	B2PEDT	B2GRPH	2	Preliminary Edit	PEDT	1	.	22JAN03
10	Book 2	.	2	B2REV	B2CEDT	2	Revise Book	REV	1	.	19FEB03
11	Book 2	.	2	B2GRPH	B2CEDT	3	Graphics	GRPH	.	1	05FEB03
12	Book 2	.	2	B2CEDT	B2PRF	1	Copyedit Book	CEDT	1	.	05MAR03
13	Book 2	.	2	B2PRF	B2PRNT	1	Proofread Book	PRF	1	.	12MAR03
14	Book 2	.	2	B2PRNT		2	Print Book	PRNT	.	.	19MAR03
15		9	1	Book 1		.			.	.	01JAN03
16		10	1	Book 2		.			.	.	22JAN03
17		13	0			.			.	.	01JAN03

Obs	S_FINISH	E_START	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	07JAN03	01JAN03	07JAN03	08JAN03	14JAN03	0		
2	07JAN03	01JAN03	07JAN03	08JAN03	14JAN03	0		
3	21JAN03	08JAN03	21JAN03	22JAN03	04FEB03	0		
4	28JAN03	08JAN03	28JAN03	15JAN03	04FEB03	0		
5	11FEB03	29JAN03	04FEB03	05FEB03	11FEB03	1	editor	
6	18FEB03	05FEB03	11FEB03	12FEB03	18FEB03	0		
7	04MAR03	12FEB03	25FEB03	19FEB03	04MAR03	0		
8	04FEB03	01JAN03	14JAN03	01JAN03	14JAN03	3	editor	
9	04FEB03	01JAN03	14JAN03	01JAN03	14JAN03	3	editor	
10	04MAR03	15JAN03	28JAN03	22JAN03	04FEB03	2	editor	
11	25FEB03	15JAN03	04FEB03	15JAN03	04FEB03	0		
12	11MAR03	05FEB03	11FEB03	05FEB03	11FEB03	0		
13	18MAR03	12FEB03	18FEB03	12FEB03	18FEB03	0		
14	01APR03	19FEB03	04MAR03	19FEB03	04MAR03	0		
15	04MAR03	01JAN03	25FEB03	08JAN03	04MAR03	0		
16	01APR03	01JAN03	04MAR03	01JAN03	04MAR03	3		
17	01APR03	01JAN03	04MAR03	01JAN03	04MAR03	0		

Output 3.6.4 *continued*

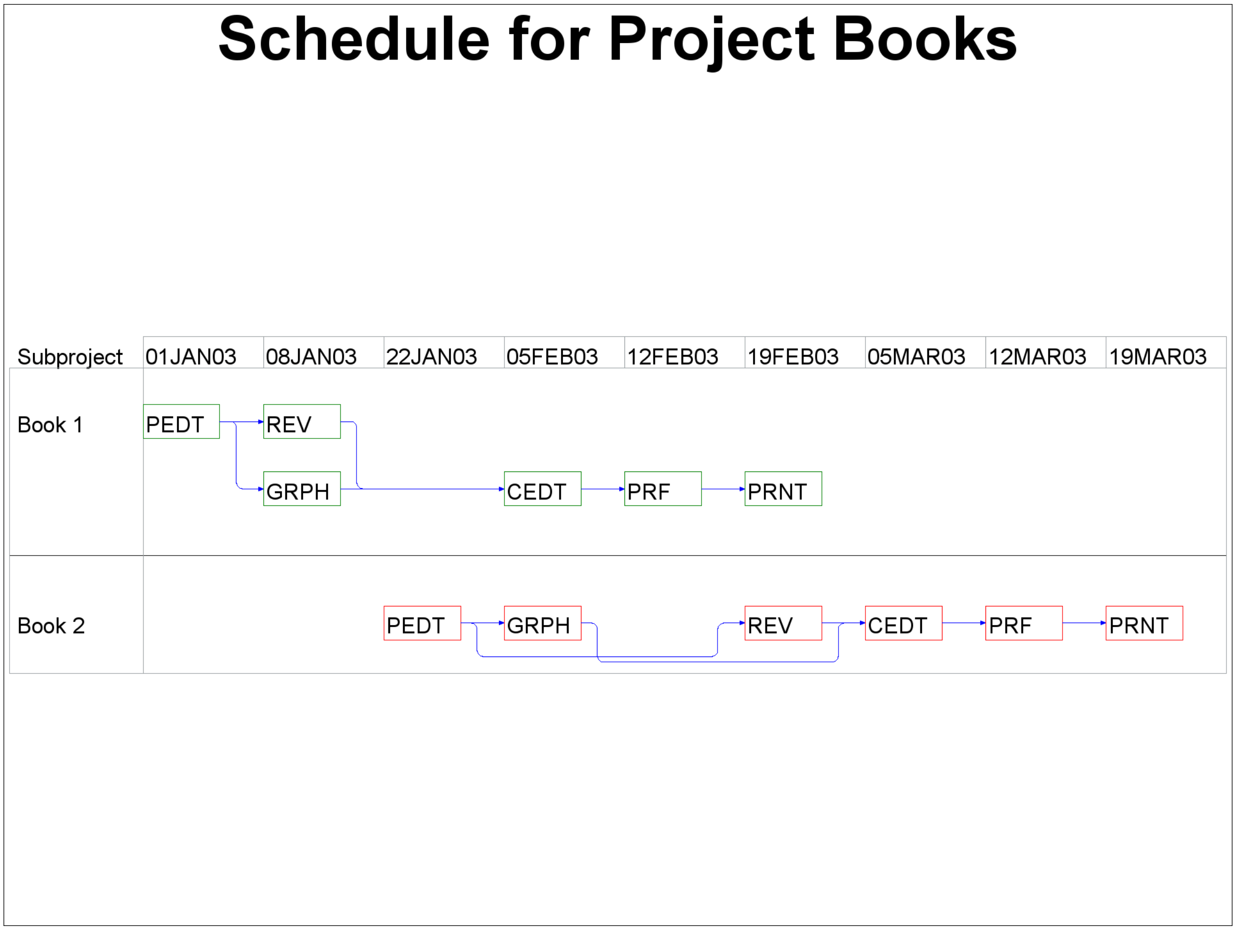
Resource Usage for Project BOOKS

Obs	_TIME_	Reditor	Aeditor	Rartist	Aartist
1	01JAN03	1	0	0	1
2	08JAN03	1	0	1	0
3	15JAN03	1	0	1	0
4	22JAN03	1	0	1	0
5	29JAN03	1	0	0	1
6	05FEB03	1	0	1	0
7	12FEB03	1	0	1	0
8	19FEB03	1	0	1	0
9	26FEB03	1	0	0	1
10	05MAR03	1	0	0	1
11	12MAR03	1	0	0	1
12	19MAR03	0	1	0	1
13	26MAR03	0	1	0	1
14	02APR03	0	1	0	1

The output data sets bookschd and booksout can be used to produce graphical reports of the schedule and the resource usage. In particular, the Schedule data set can be used to produce a zoned, time-scaled network diagram as shown in Output 3.6.5. The program used to produce the network diagram is shown in the following code. In this example, only the leaf tasks (those without any subtasks) are used to draw the network diagram. Further, the activities are aligned according to the resource-constrained start times and grouped according to the subproject.

```
goptions hpos=98 vpos=60;

pattern1 v=e c=green;
pattern2 v=e c=red;

title c=black h=4 'Schedule for Project Books';
proc netdraw data=bookschd(where=(proj_dur=.)) graphics;
   actnet / act=act succ=succ
            id=(task) nodefid nolabel
            xbetween=8 htext=3 pcompress
            zone=subproj zonepat zonespace
            align=s_start separatearcs;
   label subproj = 'Subproject';
   run;
```

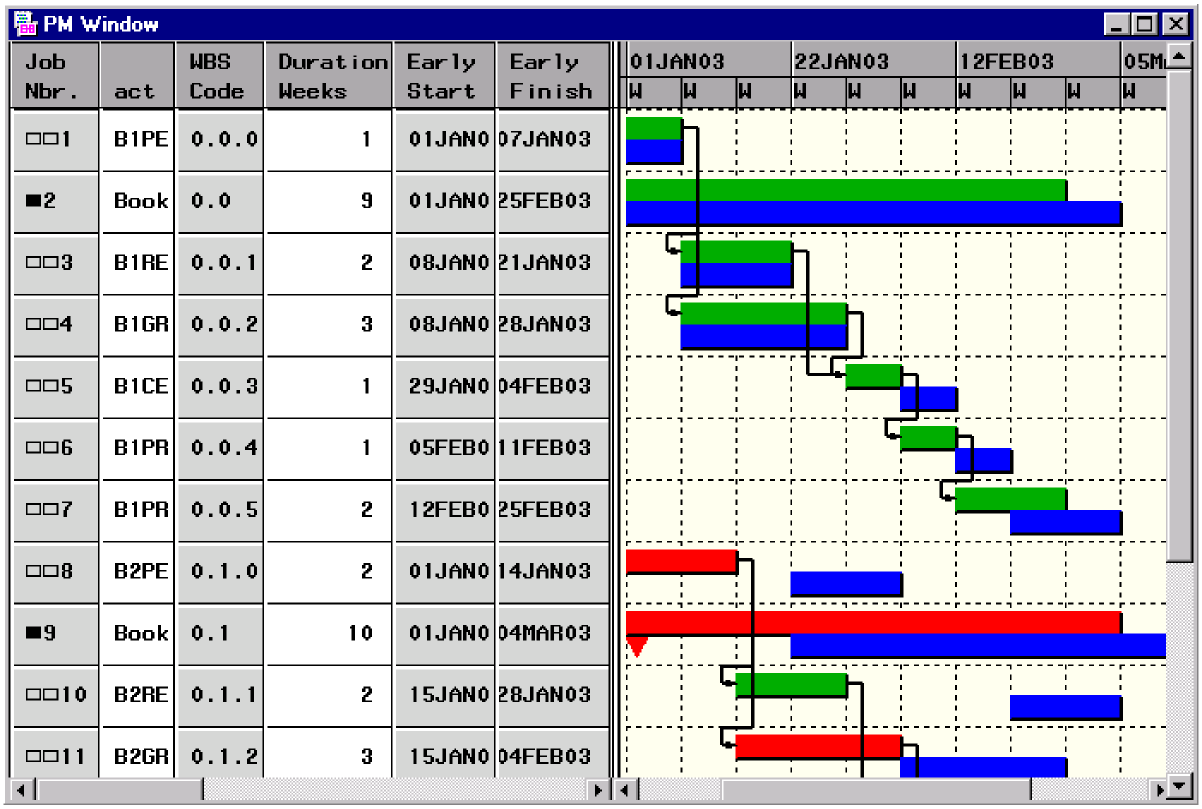
Output 3.6.5 Resource Constrained Schedule for Project Books



The same project can also be scheduled using the PM procedure, as shown in the following statements. The resulting PM Window is shown in Output 3.6.6. The advantage with using PROC PM is that you can use the PM Window to edit the activity information, such as the durations, resource requirements, and so forth.

```
proc pm data=books resin=resource
        out=pmsched resout=pmrout
        date='1jan03'd interval=week;
   act      act;
   dur      dur;
   succ     succ;
   resource editor artist / per=avdate
                            avp rcp
                            rule=actprty
                            actprty=priority
                            delayanalysis;
   id       id task;
   project  subproj;
   run;
```

Output 3.6.6 PM Window on Book Project



Example 3.7: Sequential Scheduling of Projects

Suppose the schedule displayed in Output 3.6.4 is not acceptable; you want the first book to be finished as soon as possible and do not want resources to be claimed by the second book at the cost of the first book. One way to accomplish this is to enable activities related to the second book to be split whenever the first book demands a resource currently in use by the second book. If you do not want activities to be split, you can still accomplish your goal by sequential scheduling. The structure of the input and output data sets enables you to schedule the two subprojects sequentially.

This example illustrates the sequential scheduling of subprojects ‘Book 1’ and ‘Book 2.’ The following program first schedules the subproject ‘Book 1’ using the resources available. The resulting schedule is displayed in Output 3.7.1. The Usage data set bk1out is also displayed in Output 3.7.1.

```
/* Schedule the higher priority project first */
proc cpm data=book1 resin=resource
         out=bk1schd resout=bk1out
         date='1jan03'd interval=week;
   act      act;
   dur      dur;
   succ     succ;
   resource editor artist / per=avdate avp rcp;
   id       id;
   run;
```

Output 3.7.1 Sequential Scheduling of Subprojects: Book 1

Schedule for sub-project BOOK1

Obs	act	succ	dur	id	editor	artist	S_START	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
1	B1PEDT	B1REV	1	Preliminary Edit	1	.	01JAN03	07JAN03	01JAN03	07JAN03	01JAN03	07JAN03
2	B1PEDT	B1GRPH	1	Preliminary Edit	1	.	01JAN03	07JAN03	01JAN03	07JAN03	01JAN03	07JAN03
3	B1REV	B1CEDT	2	Revise Book	1	.	08JAN03	21JAN03	08JAN03	21JAN03	15JAN03	28JAN03
4	B1GRPH	B1CEDT	3	Graphics	.	1	08JAN03	28JAN03	08JAN03	28JAN03	08JAN03	28JAN03
5	B1CEDT	B1PRF	1	Copyedit Book	1	.	29JAN03	04FEB03	29JAN03	04FEB03	29JAN03	04FEB03
6	B1PRF	B1PRNT	1	Proofread Book	1	.	05FEB03	11FEB03	05FEB03	11FEB03	05FEB03	11FEB03
7	B1PRNT		2	Print Book	.	.	12FEB03	25FEB03	12FEB03	25FEB03	12FEB03	25FEB03

Resource Usage for sub-project BOOK1

Obs	_TIME_	Reditor	Aeditor	Rartist	Aartist
1	01JAN03	1	0	0	1
2	08JAN03	1	0	1	0
3	15JAN03	1	0	1	0
4	22JAN03	0	1	1	0
5	29JAN03	1	0	0	1
6	05FEB03	1	0	0	1
7	12FEB03	0	1	0	1
8	19FEB03	0	1	0	1
9	26FEB03	0	1	0	1

The Usage data set produced by PROC CPM has two variables, Aeditor and Aartist, showing the availability of the editor and the artist on each day of the project, *after* scheduling subproject ‘Book 1.’ This data set is used to create the data set remres, listing the remaining resources available, which is then used as the Resource input data set for scheduling the subproject ‘Book 2.’ The following program shows the DATA step and the invocation of PROC CPM.

The schedule for publishing ‘Book 2’ is displayed in Output 3.7.2. The Usage data set bk2out is also displayed in Output 3.7.2. Note that this method of scheduling has ensured that ‘Book 1’ is not delayed; however, the entire project has been delayed by two more weeks, resulting in a total delay of six weeks.

```
/* Construct the Resource availability data set */
/* with proper resource names                   */
data remres;
   set bk1out;
   avdate=_time_;
   editor=aeditor;
   artist=aartist;
   keep avdate editor artist;
   format avdate date7.;
   run;

proc cpm data=book2 resin=remres
         out=bk2schd resout=bk2out
         date='1jan03'd interval=week;
   act      act;
   dur      dur;
```

```
succ     succ;
resource editor artist / per=avdate avp rcp;
id       id;
run;
```

Output 3.7.2 Sequential Scheduling of Subprojects: Book 2

Schedule for sub-project BOOK2

Obs	act	succ	dur	id	editor	artist	S_START	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
1	B2PEDT	B2REV	2	Preliminary Edit	1	.	12FEB03	25FEB03	01JAN03	14JAN03	01JAN03	14JAN03
2	B2PEDT	B2GRPH	2	Preliminary Edit	1	.	12FEB03	25FEB03	01JAN03	14JAN03	01JAN03	14JAN03
3	B2REV	B2CEDT	2	Revise Book	1	.	26FEB03	11MAR03	15JAN03	28JAN03	22JAN03	04FEB03
4	B2GRPH	B2CEDT	3	Graphics	.	1	26FEB03	18MAR03	15JAN03	04FEB03	15JAN03	04FEB03
5	B2CEDT	B2PRF	1	Copyedit Book	1	.	19MAR03	25MAR03	05FEB03	11FEB03	05FEB03	11FEB03
6	B2PRF	B2PRNT	1	Proofread Book	1	.	26MAR03	01APR03	12FEB03	18FEB03	12FEB03	18FEB03
7	B2PRNT		2	Print Book	.	.	02APR03	15APR03	19FEB03	04MAR03	19FEB03	04MAR03

Resource Usage for sub-project BOOK2

Obs	_TIME_	Reditor	Aeditor	Rartist	Aartist
1	12FEB03	1	0	0	1
2	19FEB03	1	0	0	1
3	26FEB03	1	0	1	0
4	05MAR03	1	0	1	0
5	12MAR03	0	1	1	0
6	19MAR03	1	0	0	1
7	26MAR03	1	0	0	1
8	02APR03	0	1	0	1
9	09APR03	0	1	0	1
10	16APR03	0	1	0	1

Example 3.8: Project Cost Control

Cost control and accounting are important aspects of project management. Cost data for a project may be associated with activities or groups of activities, or with resources, such as personnel or equipment. For example, consider a project that consists of several subprojects, each of which is contracted to a different company. From the contracting company's point of view, each subproject can be treated as one cost item; all the company needs to know is how much each subproject is going to cost. On the other hand, another project may contain several activities, each of which requires two types of labor, skilled and unskilled. The cost for each activity in the project may have to be computed on the basis of how much skilled or unskilled labor that activity uses. In this case, activity and project costs are determined from the resources used. Further, for any project, there may be several ways in which costs need to be summarized and accounted for. In addition to determining the cost of each individual activity, you may want to determine periodic budgets for different departments that are involved with the project or compare the actual costs that were incurred with the budgeted costs.

It is easy to set up cost accounting systems using the output data sets produced by PROC CPM, whether costs are associated with activities or with resources. In fact, you can even treat cost as a consumable resource if you can estimate the cost per day for each of the activities (see Chapter 4, "The CPM Procedure," for details about resource allocation and types of resources). This example illustrates such a method for monitoring costs and shows how you can compute some of the standard cost performance measures used in project management.

The following three measures can be used to determine if a project is running on schedule and within budget (see Moder, Phillips, and Davis 1983, for a detailed discussion on project cost control):

- *Actual cost of work performed* (**ACWP**) is the actual cost expended to perform the work accomplished in a given period of time.
- *Budgeted cost of work performed* (**BCWP**) is the budgeted cost of the work *completed* in a given period of time.
- *Budgeted cost of work scheduled* (**BCWS**) is the budgeted cost of the work *scheduled* to be accomplished in a given period of time (if a baseline schedule were followed).

Consider the survey example described earlier in this chapter. Suppose that it is possible to estimate the cost per day for each activity in the project. The following data set survcost contains the project data (activity, succ1–succ3, id, duration) and a variable named cost containing the cost per day in dollars. In order to compute the BCWS for the project, you need to establish a baseline schedule. Suppose the early start schedule computed by PROC CPM is chosen as the baseline schedule. The Resource data set costavl establishes cost as a consumable resource, so that the CPM procedure can be used to accumulate costs (using the CUMUSAGE option).

The following program invokes PROC CPM with the RESOURCE statement and saves the Usage data set in survrout. The variable ecost in this Usage data set contains the cumulative expense incurred for the baseline schedule; this is the same as the budgeted cost of work scheduled (or BCWS) saved in the data set basecost.

```
data survcost;
   format id $20. activity $8. succ1-succ3 $8. ;
   input id & activity & duration succ1 & succ2 & succ3 & cost;
   datalines;
Plan Survey           plan sur   4 hire per  design q  .         300
Hire Personnel        hire per   5 trn per   .         .         350
Design Questionnaire  design q   3 trn per   select h  print q   100
Train Personnel       trn per    3 cond sur  .         .         500
Select Households     select h   3 cond sur  .         .         300
Print Questionnaire   print q    4 cond sur  .         .         250
Conduct Survey        cond sur  10 analyze   .         .         200
Analyze Results       analyze    6 .         .         .         500
;

data holidata;
   format hol date7.;
   hol = '4jul03'd;
   run;
```

```
data costavl;
   input per & date7. otype $ cost;
   format per date7.;
   datalines;
.       restype   2
1jul03  reslevel  12000
;

proc cpm date='1jul03'd interval=weekday
         data=survcost resin=costavl   holidata=holidata
         out=sched     resout=survrout;
   activity   activity;
   successor  succ1-succ3;
   duration   duration;
   holiday    hol;
   id         id;
   resource   cost / period  = per
                     obstype = otype cumusage;
   run;

data basecost (keep = _time_ bcws);
   set survrout;
   bcws = ecost;
   run;
```

Suppose that the project started as planned on July 1, 2003, but some of the activities took longer than planned and some of the cost estimates were found to be incorrect. The following data set, actual, contains updated information: the variables as and af contain the actual start and finish times of the activities that have been completed or are in progress. The variable actcost contains the revised cost per day for each activity. The following program combines this information with the existing project data and saves the result in the data set update, displayed in Output 3.8.1. The Resource data set costavl2 (also displayed in Output 3.8.1) defines cost and actcost as consumable resources.

```
data actual;
   format id $20. ;
   input id & as & date9. af & date9. actcost;
   format as af date7.;
   datalines;
Plan Survey             1JUL03   8JUL03   275
Hire Personnel          9JUL03  15JUL03   350
Design Questionnaire  10JUL03  14JUL03   150
Train Personnel       16JUL03  17JUL03   800
Select Households     15JUL03  17JUL03   450
Print Questionnaire   15JUL03  18JUL03   250
Conduct Survey        21JUL03   .        200
;

data work.update;
   merge survcost actual;
   run;

data costavl2;
   input per & date7. otype $ cost actcost;
```

```
   format per date7.;
   datalines;
.        restype    2      2
1jul03   reslevel   12000  12000
;

title 'Activity Data Set UPDATE';
proc print data=work.update;
   run;

title 'Resource Data Set COSTAVL2';
proc print data=costavl2;
   run;
```

Output 3.8.1 Project Cost Control: Progress Update

Activity Data Set UPDATE

Obs	id	activity	succ1	succ2	succ3	duration	cost	as	af	actcost
1	Plan Survey	plan sur	hire per	design q		4	300	01JUL03	08JUL03	275
2	Hire Personnel	hire per	trn per			5	350	09JUL03	15JUL03	350
3	Design Questionnaire	design q	trn per	select h	print q	3	100	10JUL03	14JUL03	150
4	Train Personnel	trn per	cond sur			3	500	16JUL03	17JUL03	800
5	Select Households	select h	cond sur			3	300	15JUL03	17JUL03	450
6	Print Questionnaire	print q	cond sur			4	250	15JUL03	18JUL03	250
7	Conduct Survey	cond sur	analyze			10	200	21JUL03	.	200
8	Analyze Results	analyze				6	500	.	.	.

Resource Data Set COSTAVL2

Obs	per	otype	cost	actcost
1	.	restype	2	2
2	01JUL03	reslevel	12000	12000

Next, PROC CPM is used to revise the schedule by using the ACTUAL statement to specify the actual start and finish times and the RESOURCE statement to specify both the budgeted and the actual costs. The resulting schedule is saved in the data set updsched (displayed in Output 3.8.2) and the budgeted and the actual cumulative costs of the project (until the current date) are saved in the data set updtrout. These cumulative costs represent the budgeted cost of work performed (BCWP) and the actual cost of work performed (ACWP), respectively, and are saved in the data set updtcost. The two data sets basecost and updtcost are then merged to create a data set that contains the three measures: bcws, bcwp, and acwp. The resulting data set is displayed in Output 3.8.3.

```
proc cpm date='1jul03'd interval=weekday
         data=work.update    resin=costavl2
         out=updsched   resout=updtrout
         holidata=holidata;
   activity   activity;
   successor  succ1-succ3;
   duration   duration;
   holiday    hol;
   id         id;
```

```
   resource   cost actcost / per     = per
                             obstype = otype
                             maxdate = '21jul03'd cumusage;
   actual / a_start=as a_finish=af;
   run;

title 'Updated Schedule: Data Set UPDSCHED';
proc print data=updsched;
   run;

data updtcost (keep = _time_ bcwp acwp);
   set updtrout;
   bcwp = ecost;
   acwp = eactcost;
   run;

/* Create a combined data set to contain the BCWS, BCWP, ACWP */
/* per day and the cumulative values for these costs.         */
data costs;
   merge basecost updtcost;
   run;

title 'BCWS, BCWP, and ACWP';
proc print data=costs;
   run;
```

Output 3.8.2 Project Cost Control: Updated Schedule

Updated Schedule: Data Set UPDSCHED

Obs	activity	succ1	succ2	succ3	duration	STATUS	A_DUR	id	cost	actcost
1	plan sur	hire per	design q		4	Completed	5	Plan Survey	300	275
2	hire per	trn per			5	Completed	5	Hire Personnel	350	350
3	design q	trn per	select h	print q	3	Completed	3	Design Questionnaire	100	150
4	trn per	cond sur			3	Completed	2	Train Personnel	500	800
5	select h	cond sur			3	Completed	3	Select Households	300	450
6	print q	cond sur			4	Completed	4	Print Questionnaire	250	250
7	cond sur	analyze			10	In Progress	.	Conduct Survey	200	200
8	analyze				6	Pending	.	Analyze Results	500	.

Obs	A_START	A_FINISH	S_START	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
1	01JUL03	08JUL03	01JUL03	08JUL03	01JUL03	08JUL03	01JUL03	08JUL03
2	09JUL03	15JUL03	09JUL03	15JUL03	09JUL03	15JUL03	09JUL03	15JUL03
3	10JUL03	14JUL03	10JUL03	14JUL03	10JUL03	14JUL03	10JUL03	14JUL03
4	16JUL03	17JUL03	16JUL03	17JUL03	16JUL03	17JUL03	16JUL03	17JUL03
5	15JUL03	17JUL03	15JUL03	17JUL03	15JUL03	17JUL03	15JUL03	17JUL03
6	15JUL03	18JUL03	15JUL03	18JUL03	15JUL03	18JUL03	15JUL03	18JUL03
7	21JUL03	.	21JUL03	01AUG03	21JUL03	01AUG03	21JUL03	01AUG03
8	.	.	04AUG03	11AUG03	04AUG03	11AUG03	04AUG03	11AUG03

Output 3.8.3 Project Cost Control: BCWS, BCWP, ACWP

BCWS, BCWP, and ACWP

Obs	_TIME_	bcws	bcwp	acwp
1	01JUL03	0	0	0
2	02JUL03	300	300	275
3	03JUL03	600	600	550
4	07JUL03	900	900	825
5	08JUL03	1200	1200	1100
6	09JUL03	1650	1500	1375
7	10JUL03	2100	1850	1725
8	11JUL03	2550	2300	2225
9	14JUL03	3450	2750	2725
10	15JUL03	4350	3200	3225
11	16JUL03	5400	4100	4275
12	17JUL03	6150	5150	5775
13	18JUL03	6650	6200	7275
14	21JUL03	6850	6450	7525
15	22JUL03	7050	.	.
16	23JUL03	7250	.	.
17	24JUL03	7450	.	.
18	25JUL03	7650	.	.
19	28JUL03	7850	.	.
20	29JUL03	8050	.	.
21	30JUL03	8250	.	.
22	31JUL03	8450	.	.
23	01AUG03	8650	.	.
24	04AUG03	9150	.	.
25	05AUG03	9650	.	.
26	06AUG03	10150	.	.
27	07AUG03	10650	.	.
28	08AUG03	11150	.	.
29	11AUG03	11650	.	.

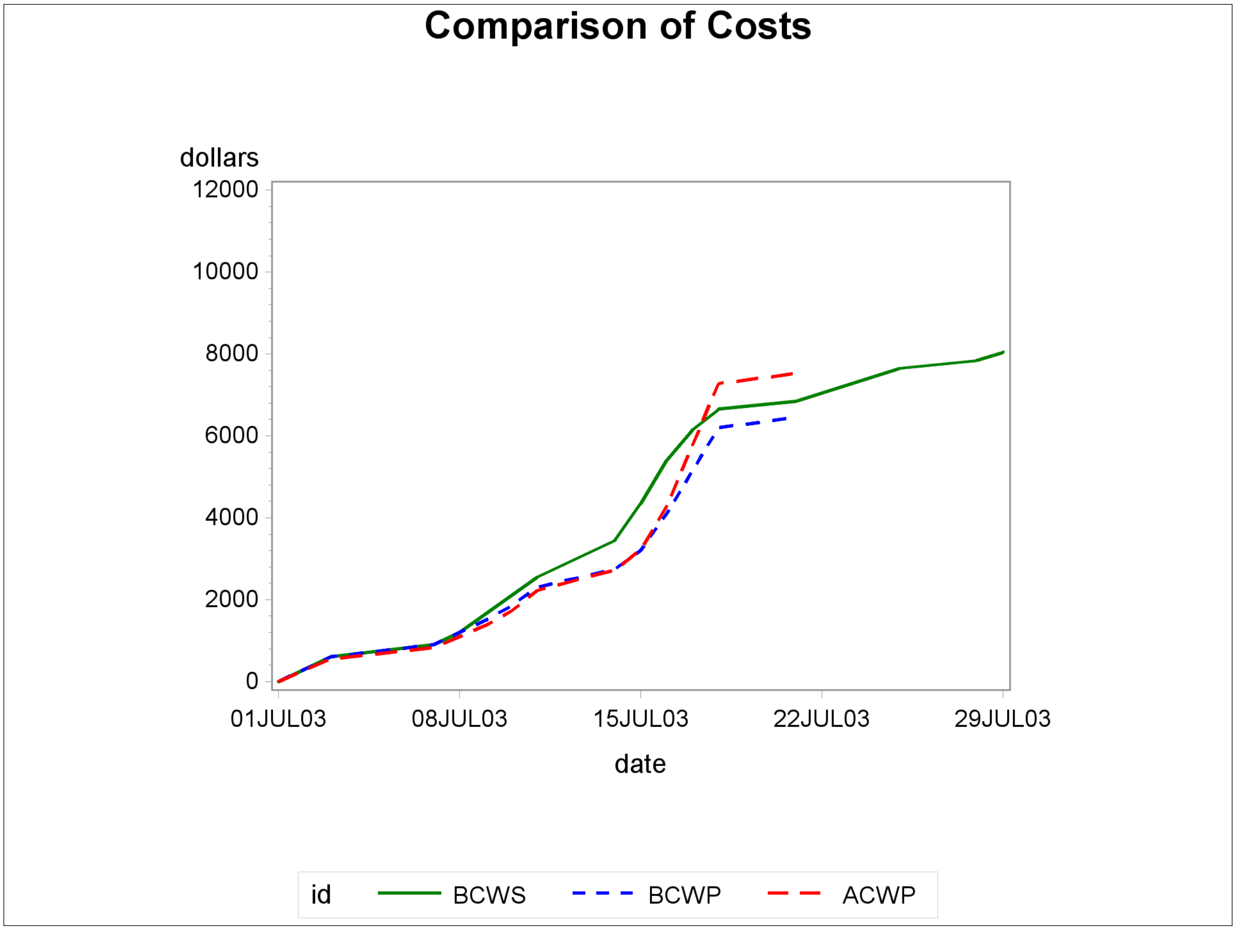
The data set costs, containing the required cost information, is then used as input to PROC GPLOT to produce a plot of the three cumulative cost measures. The plot is shown in Output 3.8.4.

NOTE: BCWS, BCWP, and ACWP are three of the cost measures used as part of *Earned Value Analysis*, which is an important component of the *Cost/Schedule Control Systems Criteria* (referred to as C/SCSC) that was established in 1967 by the Department of Defense (DOD) to standardize the reporting of cost and schedule performance on major contracts. Refer to Fleming (1988) for a detailed discussion of C/SCSC. Similar methods, such as the ones described in this example, can be used to calculate all the relevant measures for analyzing cost and schedule performance.

```
/* Plot the cumulative costs */
data costplot (keep=date dollars id);
   set costs;
   format date date7.;
   date = _time_;
   if bcws ^= . then do;
```

```
         dollars = BCWS; id = 1; output;
         end;
      if bcwp ^= . then do;
         dollars = BCWP; id = 2; output;
         end;
      if acwp ^= . then do;
         dollars = ACWP; id = 3; output;
         end;
      run;

legend1 frame
        value=(c=black j=l 'BCWS' 'BCWP' 'ACWP')
        label=(c=black);

axis1 width=2
      order=('1jul03'd to '1aug03'd by week)
      length=60 pct
      value=(c=black)
      label=(c=black);

axis2 width=2
      order=(0 to 12000 by 2000)
      length = 55 pct
      value=(c=black)
      label=(c=black);

symbol1 i=join v=none c=green  w=4 l=1;
symbol2 i=join v=none c=blue   w=4 l=2;
symbol3 i=join v=none c=red    w=4 l=3;
title c=black h=2.5 'Comparison of Costs';

proc gplot data=costplot;
   plot dollars * date = id / legend=legend1
                              haxis=axis1
                              vaxis=axis2;
   run;
```

Output 3.8.4 Plot of BCWS, BCWP, and ACWP



Example 3.9: Subcontracting Decisions

Making decisions about subcontracting forms an important part of several medium-to-large scale projects. For example, in the pharmaceutical industry, the analysis of clinical trials may be a part of the drug development project that could either be accomplished by the company's statistical group or be subcontracted to a statistical consulting firm. The decision may hinge upon how busy the local statistical group is with other projects that may delay the results of the analysis for the drug in question. Further, there may be more than one firm that is a likely candidate for performing the analysis. As a prerequisite for deciding whether to assign the *analysis* subproject to an external firm, you need to obtain a *bid* in the form of estimates of the cost and project duration from the competing firms as well as a corresponding estimate from the in-house team.

The cost corresponding to each possible subcontracting firm may be a combination of the actual costs (consulting fees and so on) and the tardiness of the project (tardiness being measured as the time difference between when the results are expected to be available and the target date for the availability of the results). The information required could be provided in terms of Gantt charts and cost analysis charts. Using this information, the project manager for the drug development project can use the principles of decision analysis to determine whether to do the analysis in-house or assign it to an outside consulting firm and to pick the firm to which the subcontract is to be assigned. Some of these ideas are illustrated in the following example.

Output 3.9.1 Input Data Sets for Decision Problem

Subcontracting Decision
The Stage Data Set

Obs	_STNAME_	_STTYPE_	_OUTCOM_	_REWARD_	_SUCCES_
1	Assignment	D	In_House	.	Complete
2			Consult1	-20,000	Act_Finish
3			Consult2	-17,500	Act_Finish
4	Complete	C	On_Time	.	Cost
5			Delay	-10,000	Cost
6	Act_Finish	C	Early	.	
7			Late	.	
8			Delay2	-1,000	
9	Cost	C	High	.	
10			Low	.	

Subcontracting Decision
The Probability Data Set

Obs	_GIVEN_	_EVENT_	_PROB_
1		High	0.50
2		Low	0.50
3		On_Time	0.60
4		Delay	0.40
5	Consult1	Early	0.60
6	Consult1	Late	0.35
7	Consult1	Delay2	0.05
8	Consult2	Early	0.50
9	Consult2	Late	0.40
10	Consult2	Delay2	0.10

Subcontracting Decision
The Payoffs Data Set

Obs	_STATE1	_STATE2	_VALUE_
1	On_Time	High	-12,000
2	On_Time	Low	-9,500
3	Delay	High	-15,000
4	Delay	Low	-11,500
5	Early		3,500
6	Late		1,500
7	Delay2		0

The stages of the decision problem are identified by the STAGEIN= data set, stage, displayed in Output 3.9.1. As a first step, the drug company needs to decide whether to perform the analysis in-house or to assign it to one of two consulting firms. If the in-house team is chosen, the resulting stage is a chance node, called 'Complete,' with two possible outcomes: 'On-Time' or 'Delay'; if there is a delay, the resulting cost to the drug company is $\$10,000$. For each of these two outcomes, there is a second chance event corresponding to the cost of the analysis. For each of the two consulting firms, the outcome can be one of three possibilities: 'Early,' 'Late,' or 'Delay2'; if there is a delay, the drug company imposes a delay penalty of $\$9,000$ on the firm, resulting in a net reward of $-\$1,000$ (penalty of $\$9,000$ minus the cost of $\$10,000$).

The PROBIN= data set, prob, identifies the various probabilities associated with the different possible outcomes at each of the chance events. The prob data set is also displayed in Output 3.9.1.

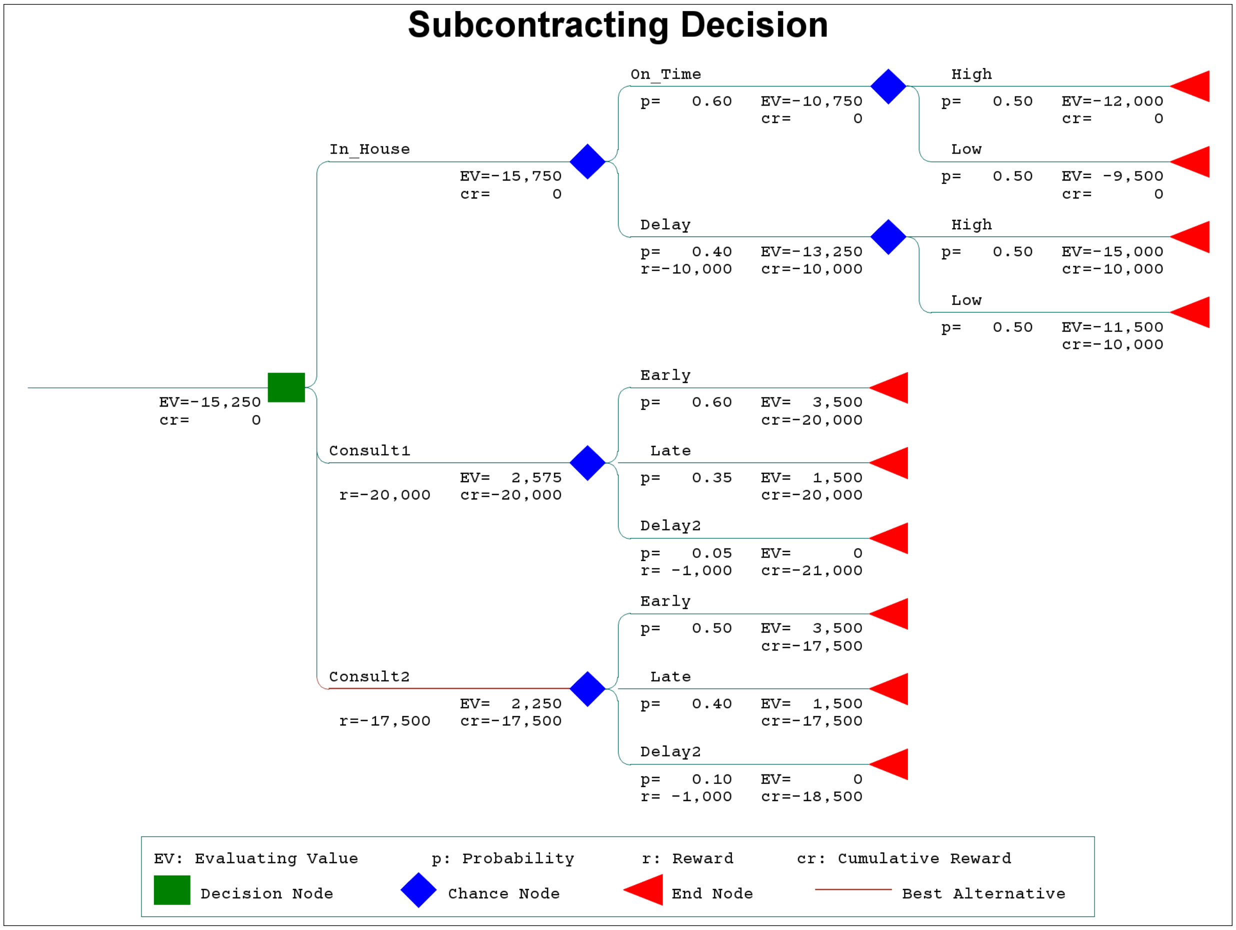
The rewards (or payoffs) associated with each of the end stages are listed in the PAYOFFS= data set, payoff (also listed in Output 3.9.1). For example, for the in-house team, the high (low) cost associated with completing the analysis on time is $\$12,000$ ($\$9,500$), and so on.

The following program invokes PROC DTREE to solve the decision problem. The complete decision tree, displayed in Output 3.9.2, represents the various stages and outcomes of the problem and identifies the optimal decision. In this example, the drug company should award the consulting contract to the second consulting firm as indicated by the dashed line for the corresponding branch of the tree.

See Chapter 7, "The DTREE Procedure," for details about the DTREE procedure.

```
title "Subcontracting Decision";
symbol1 f=marker v=P c=blue;
symbol2 f=marker v=U c=green;
symbol3 f=marker v=A c=red;

 /* PROC DTREE statements */
proc dtree stagein=stage
           probin=prob
           payoffs=payoff
           nowarning
           ;
   evaluate;
   treeplot / graphics font='Cumberland AMT'
              compress ybetween=1 cell
              lwidth=1 lwidthb=2 hsymbol=2
              symbolc=1 symbold=2 symbole=3
              lstyleb=1
              ;
quit;
```

Output 3.9.2 Decision Analysis



Project Management Systems

As illustrated in the section "Data Flow" on page 10 and the section "Examples" on page 17, the procedures of SAS/OR software, when combined with the other parts of the SAS System, provide a rich environment for developing customized project management systems. Every company has its own set of requirements for how project data should be handled and for how costs should be accounted. The CPM, GANTT, NETDRAW, and PM procedures, together with the other reporting, summarizing, charting, and plotting procedures, are the basic building blocks that can be combined in several different ways to provide the exact structure that you need. The interactive PM procedure can be used as the primary editing interface for entering all activity information for your projects. Further, the application building tools in the SAS System can be used to cement the pieces together in a menu-driven application. You can create easy-to-use applications enabling the user to enter information continually and to obtain progress reports periodically.

Web-Based Scheduling Systems

The examples in this chapter describe several scenarios that illustrate the different ways in which the project management procedures can be used to define, manage, and monitor projects. As described in the previous sections, the SAS System can be used to create comprehensive Decision Support systems or project management systems, in particular, using the procedures described in this book. With the availability of SAS/IntrNet software, you can also create Web-based project management or scheduling systems where the browser is used to display schedules and resource usage information that is updated using the CPM procedure's scheduling engine.

Examples of such Web-based applications are available at SAS Institute's external Web site at the following url: `http://support.sas.com/sassamples/demos/supplychain/demos`. In particular, the "Enterprise-Wide Resource Management" (EWRM) demo uses several of the ideas described in this chapter and illustrated in the examples throughout this book to create an application that schedules the tasks required for the maintenance of aircraft engines at a hypothetical service facility.

NOTE: The EWRM Web demo is a client-server application driven from your desktop and running at SAS Institute in Cary, NC. You can access the demo from SAS Institute's Supply Chain Web site (`http://support.sas.com/sassamples/demos/supplychain/demos/ewrm/ewrm_index.html`). The graphs and reports in the demo have not been saved, but are calculated on demand; this means that they change dynamically as the data used to calculate them change. This demo requires Internet Explorer, version 5.0 or later.

Microsoft Project Conversion Macros

%MSPTOSAS is a SAS macro that converts Microsoft Project data to a form that is readable by the PM procedure. The macro generates the necessary SAS data sets, determines the values of the relevant options, and invokes an instance of the PM procedure with the converted project data. %SASTOMSP is a SAS macro that converts data sets used by the PM and CPM procedures into a file that is readable by Microsoft Project. For details about the macros, see Chapter 6, "The Microsoft Project Conversion Macros."

Earned Value Management Macros

Earned value refers to the amount of work that has been completed within a project to date. In earned value management (EVM), this earned value is compared to the original budget and schedule in order to measure project performance, estimate future costs, and predict the project completion date.

The earned value management macros are divided into two sets. The first set is used to analyze schedule and cost data and to derive earned value metrics. The second set is used to graphically represent the results from the first set. For details, see Chapter 10, "The Earned Value Management Macros."

References

Cohen, M. (1990). "Decision Analysis in Project Management." *PM Network* 4:37–40.

Fleming, Q. W. (1988). *Cost/Schedule Control Systems Criteria: The Management Guide to C/SCSC*. Chicago: Probus Publishing.

Moder, J. J., Phillips, C. R., and Davis, E. W. (1983). *Project Management with CPM, PERT, and Precedence Diagramming*. New York: Van Nostrand Reinhold.

SAS Institute Inc. (1993). *SAS/OR Software: Project Management Examples*. Cary, NC: SAS Institute Inc.

Williams, G. A., and Boyd, W. L. (1990). "Decision Support Systems and Project Management." *PM Network* 4:31–36.

Chapter 4
The CPM Procedure

Contents

Overview: CPM Procedure

The CPM procedure can be used for planning, controlling, and monitoring a project. A typical project consists of several activities that may have precedence and time constraints. Some of these activities may already be in progress; some of them may follow different work schedules. All of the activities may compete for scarce resources. PROC CPM enables you to schedule activities subject to all of these constraints.

PROC CPM enables you to define calendars and specify holidays for the different activities so that you can schedule around holidays and vacation periods. Once a project has started, you can monitor it by specifying current information or progress data that is used by PROC CPM to compute an updated schedule. You can compare the new schedule with a baseline (or target) schedule.

For projects with scarce resources, you can determine resource-constrained schedules. PROC CPM enables you to select from a wide variety of options so that you can control the scheduling process. Thus, you may select to delay project completion time or use supplementary levels of resources, or alternate resources, if they are available.

All project information is contained in SAS data sets. The input data sets used by PROC CPM are as follows:

- The Activity data set contains all activity-related information such as activity name, precedence information, calendar used by the activity, progress information, baseline (or target schedule) information, resource requirements, time constraints, and any other information that you want to identify with each activity.
- The Resource data set specifies resource types, resource availabilities, resource priorities, and alternate resources.
- The Workday data set and the Calendar data set together enable you to specify any type of work pattern during a week and within each day of the week.
- The Holiday data set enables you to associate standard holidays and vacation periods with each calendar.

The output data sets are as follows:

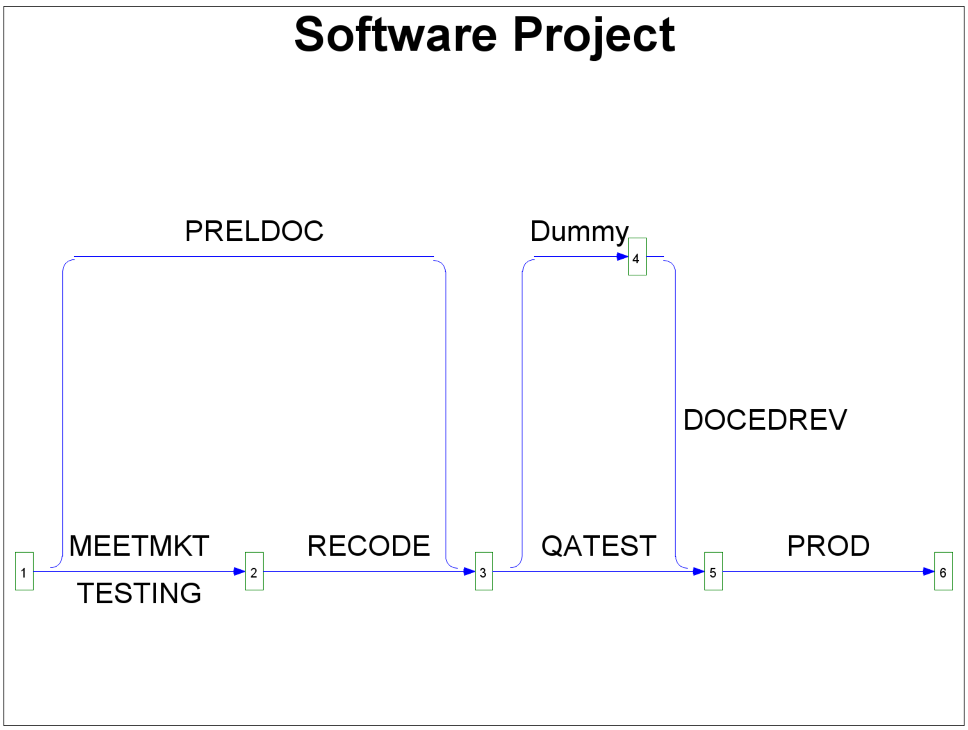
- The Schedule data set contains the early, late, baseline, resource-constrained, and actual schedules and any other activity-related information that is calculated by PROC CPM.
- The Resource Schedule data set contains the schedules for each resource used by an activity.
- The Usage data set contains the resource usage for each of the resources used in the project.

See Chapter 5, "The PM Procedure," for an interactive procedure that enables you to use a Graphical User Interface to enter and edit project information.

Getting Started: CPM Procedure

The basic steps necessary to schedule a project are illustrated using a simple example. Consider a software development project in which an applications developer has the software finished and ready for preliminary testing. In order to complete the project, several activities must take place. Certain activities cannot start until other activities have finished. For instance, the preliminary documentation must be written before it can be revised and edited and before the Quality Assurance department (QA) can test the software. Such constraints among the activities (namely, activity B can start after activity A has finished) are referred to as *precedence constraints*. Given the precedence constraints and estimated durations of the activities, you can use the *critical path method* to determine the shortest completion time for the project.

Figure 4.1 Activity-On-Arc Network



The first step in determining project completion time is to capture the relationships between the activities in a convenient representation. This is done by using a network diagram. Two types of network diagrams are popular for representing a project.

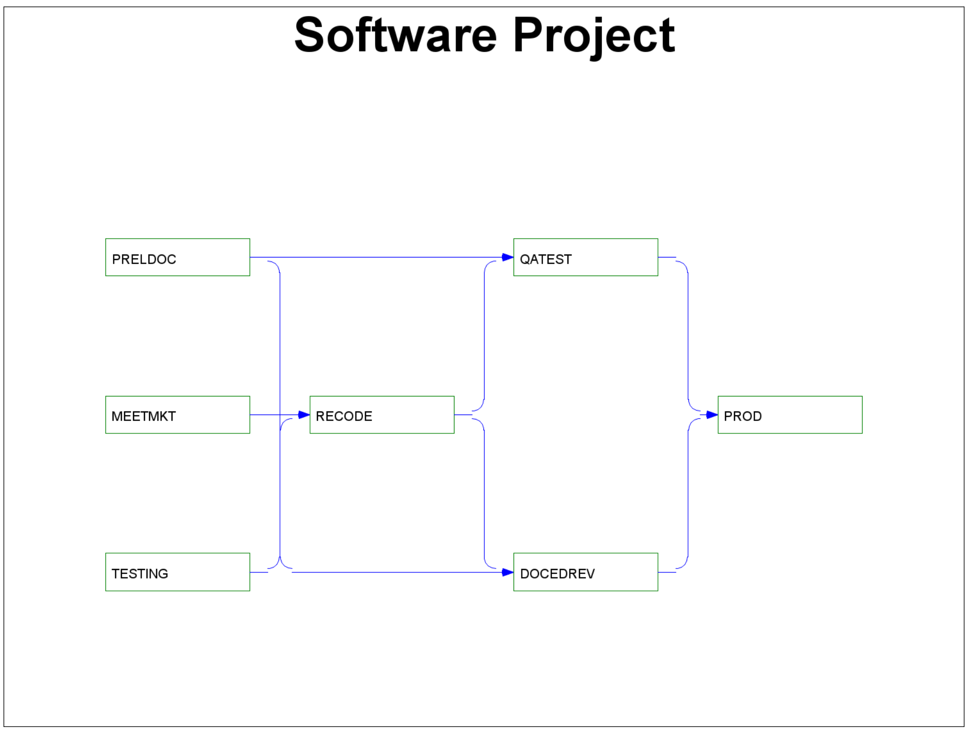
- Activity-On-Arc (AOA) or Activity-On-Edge (AOE) diagrams show the activities on the arcs or edges of the network. Figure 4.1 shows the AOA representation for the software project. This method of representing a project is known also as the arrow diagramming method (ADM). For projects represented in the AOA format, PROC CPM requires the use of the following statements:

  PROC CPM *options* ;
      TAILNODE *variable* ;
      HEADNODE *variable* ;
      DURATION *variable* ;

- Activity-On-Node (AON) or Activity-On-Vertex (AOV) diagrams show the activities on nodes or vertices of the network. Figure 4.2 shows the AON representation of the project. This method is known also as the *precedence diagramming method* (PDM). The AON representation is more flexible because it enables you to specify nonstandard precedence relationships between the activities (for example, you can specify that activity B starts five days after the start of activity A). PROC CPM requires the use of the following statements to schedule projects that are represented using the AON format:

  PROC CPM *options* **;**
      ACTIVITY *variable* **;**
      SUCCESSOR *variables* **;**
      DURATION *variable* **;**

Figure 4.2 Activity-On-Node Network



The AON representation of the network is used in the remainder of this section to illustrate some of the features of PROC CPM. The project data are input to PROC CPM using a SAS data set. The basic project information is conveyed to PROC CPM through the ACTIVITY, SUCCESSOR, and DURATION statements. Each observation of the Activity data set specifies an activity in the project, its duration, and its immediate successors. PROC CPM enables you to specify all of the immediate successors in the same observation, or you can have multiple observations for each activity, listing each successor in a separate observation. (Multiple variables in the SUCCESSOR statement are used here.) PROC CPM enables you to use long activity names. In this example, shorter names are used for the activities to facilitate data entry; a variable, Descrpt, is used to specify a longer description for each activity.

The procedure determines items such as the following:

- the minimum time in which the project can be completed
- the set of activities that is critical to the completion of the project in the minimum amount of time

No displayed output is produced. However, the results are saved in an output data set (the Schedule data set) that is shown in Figure 4.3.

The code for the entire program is as follows.

```
data software;
   format Descrpt $20. Activity $8.
          Succesr1-Succesr2 $8. ;
   input Descrpt & Duration Activity $
         Succesr1 $ Succesr2 $ ;
   datalines;
Initial Testing       20  TESTING   RECODE    .
Prel. Documentation   15  PRELDOC   DOCEDREV  QATEST
Meet Marketing        1   MEETMKT   RECODE    .
Recoding              5   RECODE    DOCEDREV  QATEST
QA Test Approve       10  QATEST    PROD      .
Doc. Edit and Revise  10  DOCEDREV  PROD      .
Production            1   PROD      .         .
;

proc cpm data=software
         out=intro1
         interval=day
         date='01mar04'd;
   id descrpt;
   activity activity;
   duration duration;
   successor succesr1 succesr2;
run;

title 'Project Schedule';
proc print data=intro1;
run;
```

Figure 4.3 Software Project Plan

Project Schedule

Obs	Activity	Succesr1	Succesr2	Duration	Descrpt
1	TESTING	RECODE		20	Initial Testing
2	PRELDOC	DOCEDREV	QATEST	15	Prel. Documentation
3	MEETMKT	RECODE		1	Meet Marketing
4	RECODE	DOCEDREV	QATEST	5	Recoding
5	QATEST	PROD		10	QA Test Approve
6	DOCEDREV	PROD		10	Doc. Edit and Revise
7	PROD			1	Production

Obs	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	01MAR04	20MAR04	01MAR04	20MAR04	0	0
2	01MAR04	15MAR04	11MAR04	25MAR04	10	10
3	01MAR04	01MAR04	20MAR04	20MAR04	19	19
4	21MAR04	25MAR04	21MAR04	25MAR04	0	0
5	26MAR04	04APR04	26MAR04	04APR04	0	0
6	26MAR04	04APR04	26MAR04	04APR04	0	0
7	05APR04	05APR04	05APR04	05APR04	0	0

In addition to the variables specified in the ACTIVITY, SUCCESSOR, DURATION, and ID statements, the output data set contains the following new variables.

E_START
: specifies the earliest time an activity can begin, subject to any time constraints and the completion time of the preceding activity.

E_FINISH
: specifies the earliest time an activity can be finished, assuming it starts at E_START.

L_START
: specifies the latest time an activity can begin so that the project is not delayed.

L_FINISH
: specifies the latest time an activity can be finished without delaying the project.

T_FLOAT
: specifies the amount of flexibility in the starting of a specific activity without delaying the project:

T_FLOAT = L_START − E_START = L_FINISH − E_FINISH

F_FLOAT
: specifies the difference between the early finish time of the activity and the early start time of the activity's immediate successors.

In Figure 4.3 the majority of the tasks have a total float value of 0. These events are *critical*; that is, any delay in these activities will cause the project to be delayed. Some of the activities have slack present, which means that they can be delayed by that amount without affecting the project completion date. For example, the activity MEETMKT has a slack period of 19 days because there are 19 days between 01MAR04 and 20MAR04.

The INTERVAL= option in the PROC CPM statement enables you to specify the durations of the activities in one of several possible units including days, weeks, months, hours, and minutes. In addition, you can schedule activities around weekends and holidays. (To skip weekends, you specify INTERVAL=WEEKDAY.) You can also select different patterns of work during a day or a week (for example, holidays on Friday and Saturday) and different sets of holidays for the different activities in the project. A *calendar* consists of a set of work schedules for a typical week and a set of holidays. PROC CPM enables you to define any number of calendars and associate different activities with different calendars.

In the previous example, you saw that you could schedule your project by selecting a project start date. You can also specify a project finish date if you have a deadline to be met and you need to determine the latest start times for the different activities in the project. You can also set constraints on start or finish dates for specific activities within a given project. For example, testing the software may have to be delayed until the testing group finishes another project that has a higher priority. PROC CPM can schedule the project subject to such restrictions through the use of the ALIGNDATE and ALIGNTYPE statements. See Example 4.12 for more information about the use of the ALIGNDATE and ALIGNTYPE statements.

For a project that is already in progress, you can incorporate the *actual* schedule of the activities (some activities may already be completed while others may still be in progress) to obtain a progress update. You can save the original schedule as a *baseline* schedule and use it to compare against the current schedule to determine if any of the activities have taken longer than anticipated.

Quite often the resources needed to perform the activities in a project are available only in limited quantities and may cause certain activities to be postponed due to unavailability of the required resources. You can use PROC CPM to schedule the activities in a project subject to resource constraints. A wide range of options enables you to control the scheduling process. For example, you can specify resource or activity priorities, set constraints on the maximum amount of delay that can be tolerated for a given activity, enable activities to be preempted, specify alternate resources that can be used instead of scarce resources, or indicate secondary levels of resources that can be used when the primary levels are insufficient.

When an activity requires multiple resources, it is possible that each resource may follow a different calendar and each may require varying amounts of work. PROC CPM enables you to define resource-driven durations for the activities. You can also specify calendars for the resources. In either of these situations it is possible that each resource used by an activity may have its own individual schedule. PROC CPM enables you to save the resource schedules for the different activities in a Resource Schedule data set, the RESOURCESCHED= data set.

In addition to obtaining a resource-constrained schedule in an output data set, you can save the resource utilization summary in another output data set, the RESOURCEOUT= data set. Several options enable you to control the amount of information saved in this data set.

The CPM procedure enables you to define activities in a multiproject environment with multiple levels of nesting. You can specify a PROJECT variable that identifies the name or number of the project to which each activity belongs.

All the options available with the CPM procedure are discussed in detail in the following sections. Several examples illustrate most of the features.

Syntax: CPM Procedure

The following statements are used in PROC CPM:

PROC CPM *options* ;
ACTIVITY *variable* ;
ACTUAL */ actual options* ;
ALIGNDATE *variable* ;
ALIGNTYPE *variable* ;
BASELINE */ baseline options* ;
CALID *variable* ;
DURATION */ duration options* ;
HEADNODE *variable* ;
HOLIDAY *variable / holiday options* ;
ID *variables* ;
PROJECT *variable / project options* ;
RESOURCE *variables / resource options* ;
SUCCESSOR *variables / lag options* ;
TAILNODE *variable* ;

Functional Summary

Table 4.1 outlines the options available for the CPM procedure, classified by function.

Table 4.1 Functional Summary

Description	Statement	Option
Activity Splitting Specifications		
Splits in-progress activities at TIMENOW	ACTUAL	TIMENOWSPLT
Specifies the maximum number of segments variable	RESOURCE	MAXNSEGMT=
Specifies the minimum segment duration variable	RESOURCE	MINSEGMTDUR=
Enables splitting	RESOURCE	SPLITFLAG
Baseline or Target Schedule Specifications		
Specifies the baseline finish date variable	BASELINE	B_FINISH=
Specifies the baseline start date variable	BASELINE	B_START=
Specifies the schedule to compare with baseline	BASELINE	COMPARE=
Specifies the schedule to use as baseline	BASELINE	SET=
Specifies the schedule to update baseline	BASELINE	UPDATE=
Calendar Specifications		
Specifies the calendar variable	CALID	
Specifies the holiday variable	HOLIDAY	
Specifies the holiday duration variable	HOLIDAY	HOLIDUR=
Specifies the holiday finish variable	HOLIDAY	HOLIFIN=

Table 4.1 *continued*

Description	Statement	Option
Data Set Specifications		
Specifies the Calendar input data set	PROC CPM	CALEDATA=
Specifies the Activity input data set	PROC CPM	DATA=
Specifies the Holiday input data set	PROC CPM	HOLIDATA=
Specifies the Schedule Output data set	PROC CPM	OUT=
Specifies the Resource Availability input data set	PROC CPM	RESOURCEIN=
Specifies the Resource Schedule output data set	PROC CPM	RESOURCESCHED=
Specifies the Resource Usage output data set	PROC CPM	RESOURCEOUT=
Specifies the Workday input data set	PROC CPM	WORKDATA=
Duration Control Specifications		
Specifies the workday length	PROC CPM	DAYLENGTH=
Specifies the workday start	PROC CPM	DAYSTART=
Specifies the duration unit	PROC CPM	INTERVAL=
Specifies the duration multiplier	PROC CPM	INTPER=
Converts milestones into finish milestones	PROC CPM	SETFINISHMILESTONE
Specifies the duration variable	DURATION	
Specifies the finish variable	DURATION	FINISH=
Overrides specified duration	DURATION	OVERRIDEDUR
Specifies the start variable	DURATION	START=
Specifies the work variable	RESOURCE	WORK=
Lag Specifications		
Specifies the name of the lag duration calendar	SUCCESSOR	ALAGCAL=
Specifies the lag variables	SUCCESSOR	LAG=
Specifies the number of the lag duration calendar	SUCCESSOR	NLAGCAL=
Miscellaneous Options		
Suppresses warning messages	PROC CPM	SUPPRESSOBSWARN
Fixes L_FINISH for finish tasks to E_FINISH	PROC CPM	FIXFINISH
Network Specifications		
Specifies the AON format activity variable	ACTIVITY	
Specifies the AOA format headnode variable	HEADNODE	
Specifies the project variable	PROJECT	
Specifies the AON format successor variables	SUCCESSOR	
Specifies the AOA format tailnode variable	TAILNODE	
Multiproject Specifications		
Specifies the project variable	PROJECT	
Aggregates parent resources	PROJECT	AGGREGATEPARENTRES
Ignores parent resources	PROJECT	IGNOREPARENTRES
Computes separate critical paths	PROJECT	SEPCRIT
Uses specified project duration	PROJECT	USEPROJDUR

Table 4.1 *continued*

Description	Statement	Option
Computes WBS Code	PROJECT	WBSCODE
OUT= Data Set Options		
Includes percent complete variable	ACTUAL	ESTIMATEPCTC
Adds an observation for missing activities	PROC CPM	ADDACT
Specifies single observation per activity	PROC CPM	COLLAPSE
Copies relevant variables to Schedule data set	PROC CPM	XFERVARS
Specifies the variables to be copied to Schedule data set	ID	
Includes descending sort variables	PROJECT	DESCENDING
Includes all sort order variables	PROJECT	ORDERALL
Includes early start sort order variable	PROJECT	ESORDER
Includes late start sort order variable	PROJECT	LSORDER
Includes resource start order variable	PROJECT	SSORDER
Includes WBS Code	PROJECT	WBSCODE
Includes information about resource delays	RESOURCE	DELAYANALYSIS
Includes early start schedule	RESOURCE	E_START
Includes free float	RESOURCE	F_FLOAT
Sets unscheduled S_START and S_FINISH	RESOURCE	FILLUNSCHED
Includes late start schedule	RESOURCE	L_START
Excludes early start schedule	RESOURCE	NOE_START
Excludes free float	RESOURCE	NOF_FLOAT
Excludes late start schedule	RESOURCE	NOL_START
Excludes resource variables	RESOURCE	NORESOURCEVARS
Excludes total float	RESOURCE	NOT_FLOAT
Includes resource variables	RESOURCE	RESOURCEVARS
Includes total float	RESOURCE	T_FLOAT
Sets unscheduled S_START and S_FINISH to missing	RESOURCE	UNSCHEDMISS
Updates unscheduled S_START, S_FINISH	RESOURCE	UPDTUNSCHED
Problem Size Options		
Specifies the number of precedence constraints	PROC CPM	NADJ=
Specifies the number of activities	PROC CPM	NACTS=
Specifies the number of distinct node or activity names	PROC CPM	NNODES=
Specifies the number of resource requirements	PROC CPM	NRESREQ=
Disables use of the Utility data set	PROC CPM	NOUTIL
Progress Updating Options		
Specifies the actual finish variable	ACTUAL	A_FINISH=
Specifies the actual start variable	ACTUAL	A_START=
Assumes automatic completion	ACTUAL	AUTOUPDT
Enables actual time to fall in a non-work period	ACTUAL	FIXASTART
Does not assume automatic completion	ACTUAL	NOAUTOUPDT
Specifies the percentage complete variable	ACTUAL	PCTCOMP=
Specifies the remaining duration variable	ACTUAL	REMDUR=

Table 4.1 *continued*

Description	Statement	Option
Specifies that progress updating should override resource scheduling (Experimental)	RESOURCE	SETFINISH=
Shows float for all activities	ACTUAL	SHOWFLOAT
Specifies the current date	ACTUAL	TIMENOW=
Resource Variable Specifications		
Specifies the resource variables	RESOURCE	
Specifies the observation type variable	RESOURCE	OBSTYPE=
Specifies the resource availability date/time variable	RESOURCE	PERIOD=
Specifies the alternate resource variable	RESOURCE	RESID=
Specifies the work variable	RESOURCE	WORK=
Resource Allocation Control Options		
Specifies the delay variable	RESOURCE	ACTDELAY=
Specifies the activity priority variable	RESOURCE	ACTIVITYPRTY=
Uses alternate resources before supplementary levels	RESOURCE	ALTBEFORESUP
Waits until L_START + DELAY	RESOURCE	AWAITDELAY
Specifies the delay	RESOURCE	DELAY=
Schedules even if there are insufficient resources	RESOURCE	INFEASDIAGNOSTIC
Specifies independent allocation	RESOURCE	INDEPENDENTALLOC
Enables milestones to consume resources	RESOURCE	MILESTONERESOURCE
Prevents milestones from consuming resources	RESOURCE	MILESTONENORESOURCE
Uses multiple alternates for a single resource	RESOURCE	MULTIPLEALTERNATES
Specifies the resource calendar intersect	RESOURCE	RESCALINTERSECT
Specifies the scheduling priority rule	RESOURCE	SCHEDRULE=
Specifies the secondary scheduling priority rule	RESOURCE	SCHEDRULE2=
Specifies the stop date for resource constrained scheduling	RESOURCE	STOPDATE=
RESOURCEOUT= Data Set Options		
Includes all types of resource usage	RESOURCE	ALL
Appends observations for total usage	RESOURCE	APPEND
Specifies the name of the calendar for _TIME_ increment	RESOURCE	AROUTCAL=
Includes availability profile for each resource	RESOURCE	AVPROFILE
Specifies the cumulative usage for consumable resources	RESOURCE	CUMUSAGE
Includes early start profile for each resource	RESOURCE	ESPROFILE
Excludes unscheduled activities in profile	RESOURCE	EXCLUNSCHED
Includes unscheduled activities in profile	RESOURCE	INCLUNSCHED
Records total usage of resource	RESOURCE	TOTUSAGE
Includes late start profile for each resource	RESOURCE	LSPROFILE
Specifies the maximum value of _TIME_	RESOURCE	MAXDATE=
Specifies the maximum number of observations	RESOURCE	MAXOBS=
Specifies the minimum value of _TIME_	RESOURCE	MINDATE=
Specifies the numeric calendar for _TIME_	RESOURCE	NROUTCAL=

Table 4.1 *continued*

Description	Statement	Option
Includes resource constrained profile	RESOURCE	RCPROFILE
Specifies the unit of difference between consecutive _TIME_ values	RESOURCE	ROUTINTERVAL=
Specifies the difference between consecutive _TIME_ values	RESOURCE	ROUTINTPER=
Uses a continuous calendar for _TIME_	RESOURCE	ROUTNOBREAK
RESOURCESCHED= Data Set Options		
Adds activity or resource calendar	RESOURCE	ADDCAL
Includes WBS code	PROJECT	RSCHEDWBS
Includes order variables	PROJECT	RSCHEDORDER
Specifies the ID variables	RESOURCE	RSCHEDID=
Time Constraint Specifications		
Specifies the alignment date variable	ALIGNDATE	
Specifies the alignment type variable	ALIGNTYPE	
Specifies the project start date	PROC CPM	DATE=
Specifies the project finish date	PROC CPM	FBDATE=
Finishes before DATE= value	PROC CPM	FINISHBEFORE

PROC CPM Statement

PROC CPM *options* **;**

The following options can appear in the PROC CPM statement.

ADDACT

ADDALLACT

EXPAND

indicates that an observation is to be added to the Schedule output data set (and the Resource Schedule output data set) for each activity that appears as a value of the variables specified in the SUCCESSOR or PROJECT statements without appearing as a value of the variable specified in the ACTIVITY statement. If the PROJECT statement is used, and the activities do not have a single common parent, an observation is also added to the Schedule data set containing information for a single common parent defined by the procedure.

CALEDATA=*SAS-data-set*

CALENDAR=*SAS-data-set*

identifies a SAS data set that specifies the work pattern during a standard week for each of the calendars that are to be used in the project. Each observation of this data set (also referred to as the **Calendar** data set) contains the name or the number of the calendar being defined in that observation, the names of the shifts or work patterns used each day, and, optionally, a standard workday length in hours. For details about the structure of this data set, see the section "Multiple Calendars" on page 101. The work shifts referred to in the Calendar data set are defined in the Workday data set. The calendars defined in the Calendar data set can be identified with different activities in the project.

COLLAPSE

creates only one observation per activity in the output data set when the input data set for a network in AON format contains multiple observations for the same activity. This option is allowed only if the network is in AON format.

Often, the input data set may have more than one observation per activity (especially if the activity has several successors). If you are interested only in the schedule information about the activity, there is no need for multiple observations in the output data set for this activity. Use the COLLAPSE option in this case.

DATA=*SAS-data-set*

names the SAS data set that contains the network specification and activity information. If the DATA= option is omitted, the most recently created SAS data set is used. This data set (also referred to in this chapter as the **Activity** data set) contains all of the information that is associated with each activity in the network.

DATE=*date*

specifies the SAS date, time, or datetime that is to be used as an alignment date for the project. If neither the FINISHBEFORE option nor any other alignment options are specified, then the CPM procedure schedules the project to start on *date*. If *date* is a SAS time value, the value of the INTERVAL= parameter should be HOUR, MINUTE, or SECOND; if it is a SAS date value, *interval* should be DAY, WEEKDAY, WORKDAY, WEEK, MONTH, QTR, or YEAR; and if it is a SAS datetime value, *interval* should be DTWRKDAY, DTDAY, DTHOUR, DTMINUTE, DTSECOND, DTWEEK, DTMONTH, DTQTR, or DTYEAR.

DAYLENGTH=*daylength*

specifies the length of the workday. On each day, work is scheduled starting at the beginning of the day as specified in the DAYSTART= option and ending *daylength* hours later. The DAYLENGTH= value should be a SAS time value. The default value of *daylength* is 24 if the INTERVAL= option is specified as DTDAY, DTHOUR, DTMINUTE, or DTSECOND, and the default value of *daylength* is 8 if the INTERVAL= option is specified as WORKDAY or DTWRKDAY. If INTERVAL=DAY or WEEKDAY and the value of *daylength* is less than 24, then the schedule produced is in SAS datetime values. For other values of the INTERVAL= option, the DAYLENGTH= option is ignored.

DAYSTART=*daystart*

specifies the start of the workday. The DAYSTART= value should be a SAS time value. This parameter should be specified only when *interval* is one of the following: DTDAY, WORKDAY, DTWRKDAY, DTHOUR, DTMINUTE, or DTSECOND; in other words, this parameter should be specified only if the schedule produced by the CPM procedure is in SAS datetime values. The default value of *daystart* is 9 a.m. if INTERVAL is WORKDAY; otherwise, the value of *daystart* is equal to the time part of the SAS datetime value specified for the DATE= option.

FBDATE=*fbdate*

specifies a finish-before date that can be specified in addition to the DATE= option. If the FBDATE= option is not given but the FINISHBEFORE option is specified, then *fbdate* = *date*. Otherwise, *fbdate* is equal to the project completion date. If *fbdate* is given in addition to the DATE= and FINISHBEFORE options, then the minimum of the two dates is used as the required project completion date. See the section "Scheduling Subject to Precedence Constraints" on page 93 for details about how the procedure uses the *date* and *fbdate* to compute the early and late start schedules.

FINISHBEFORE

specifies that the project be scheduled to complete before the date given in the DATE= option.

FIXFINISH

specifies that all **finish** tasks are to be constrained by their respective early finish times. In other words, the late finish times of all finish tasks do not float to the project completion time.

HOLIDATA=*SAS-data-set*

HOLIDAY=*SAS-data-set*

identifies a SAS data set that specifies holidays. These holidays can be associated with specific calendars that are also identified in the HOLIDATA= data set (also referred to as the **Holiday** data set). The HOLIDATA= option must be used with a HOLIDAY statement that specifies the variable in the SAS data set that contains the start time of holidays. Optionally, the data set can include a variable that specifies the length of each holiday or a variable that identifies the finish time of each holiday (if the holidays are longer than one day). For projects involving multiple calendars, this data set can also include the variable specified by the CALID statement that identifies the calendar to be associated with each holiday. See the section "Multiple Calendars" on page 101 for further information regarding holidays and multiple calendars.

INTERVAL=*interval*

requests that each unit of duration be measured in *interval* units. Possible values for *interval* are DAY, WEEK, WEEKDAY, WORKDAY, MONTH, QTR, YEAR, HOUR, MINUTE, SECOND, DTDAY, DTWRKDAY, DTWEEK, DTMONTH, DTQTR, DTYEAR, DTHOUR, DTMINUTE, and DTSECOND. The default value is based on the format of the DATE= parameter. See the section "Using the INTERVAL= Option" on page 94 for further information regarding this option.

INTPER=*period*

requests that each unit of duration be equivalent to *period* units of duration. The default value is 1.

NACTS=*nacts*

specifies the number of activities for which memory is allocated in core by the procedure. If the number of activities exceeds *nacts*, the procedure uses a utility data set for storing the activity array. The default value for *nacts* is set to *nobs*, if the network is specified in AOA format, and to $nobs \times (nsucc+1)$, if the network is specified in AON format, where *nobs* is the number of observations in the Activity data set and *nsucc* is the number of variables specified in the SUCCESSOR statement.

NADJ=*nadj*

specifies the number of precedence constraints (adjacencies) in the project network. If the number of adjacencies exceeds *nadj*, the procedure uses a utility data set for storing the adjacency array. The default value of *nadj* is set to *nacts* if the network is in AON format, and it is set to $nacts \times 2$ if the network is in AOA format.

NNODES=*nnodes*

specifies the size of the symbolic table used to look up the activity names (node names) for the network specification in AON (AOA) format. If the number of distinct names exceeds *nnodes*, the procedure uses a utility data set for storing the tree used for the table lookup. The default value for *nnodes* is set to *nobs*$\times$2 if the network is specified in AOA format and to *nobs*$\times$(*nsucc*+1) if the network is specified in AON format, where *nobs* is the number of observations in the Activity data set and *nsucc* is the number of variables specified in the SUCCESSOR statement.

NOUTIL

specifies that the procedure should not use utility data sets for memory management. By default, the procedure resorts to the use of utility data sets and swaps between core memory and utility data sets as necessary if the number of activities or precedence constraints or resource requirements in the input data sets is larger than the number of each such entity for which memory is initially allocated in core. Specifying this option causes the procedure to increase the memory allocation instead of using a utility data set; if the problem is too large to fit in core memory, PROC CPM will stop with an error message.

NRESREQ=*nres*

specifies the number of distinct resource requirements corresponding to all activities and resources in the project. The default value of *nres* is set to *nobs*$\times$*nresvar* $\times$0.25, where *nobs* is the number of observations in the Activity data set, and *nresvar* is the number of RESOURCE variables in the Activity data set.

OUT=*SAS-data-set*

specifies a name for the output data set that contains the schedule determined by PROC CPM. This data set (also referred to as the **Schedule** data set) contains all of the variables that were specified in the Activity data set to define the project. Every observation in the Activity data set has a corresponding observation in this output data set. If PROC CPM is used to determine a schedule that is not subject to any resource constraints, then this output data set contains the early and late start schedules; otherwise, it also contains the resource-constrained schedule. See the section "OUT= Schedule Data Set" on page 99 for information about the names of the new variables in the data set. If the OUT= option is omitted, the SAS system creates a data set and names it according to the DATA*n* naming convention.

RESOURCEIN=*SAS-data-set*

RESIN=*SAS-data-set*

RIN=*SAS-data-set*

RESLEVEL=*SAS-data-set*

names the SAS data set that contains the levels available for the different resources used by the activities in the project. This data set also contains information about the type of resource (replenishable or consumable), the calendar associated with each resource, the priority for each resource, and lists, for each resource, all the alternate resources that can be used as a substitute. In addition, this data set indicates whether or not the resource rate affects the duration. The specification of the RESIN= data set (also referred to as the **Resource** data set) indicates to PROC CPM that the schedule of the project is to be determined subject to resource constraints. For further information about the format of this data set, see the section "RESOURCEIN= Input Data Set" on page 114.

If this option is specified, you must also use the RESOURCE statement to identify the variable names for the resources to be used for resource-constrained scheduling. In addition, you must specify the name of the variable in this data set (using the PERIOD= option in the RESOURCE statement) that contains the dates from which the resource availabilities in each observation are valid. Furthermore, the data set must be sorted in order of increasing values of this period variable.

RESOURCEOUT=*SAS-data-set*

RESOUT=*SAS-data-set*

ROUT=*SAS-data-set*

RESUSAGE=*SAS-data-set*

names the SAS data set in which you can save resource usage profiles for each of the resources specified in the RESOURCE statement. This data set is also referred to as the **Usage** data set. In the Usage data set, you can save the resource usage by time period for the early start, late start, and resource-constrained schedules, and the surplus level of resources remaining after resource allocation is performed.

By default, it provides the usage profiles for the early and late start schedules if resource allocation is not performed. If resource allocation is performed, this data set also provides usage profiles for the resource-constrained schedule and a profile of the level of remaining resources.

You can control the types of profiles to be saved by using the ESPROFILE (early start usage), LSPROFILE (late start usage), RCPROFILE (resource-constrained usage), or AVPROFILE (resource availability after resource allocation) options in the RESOURCE statement. You can specify any combination of these four options. You can also specify the ALL option to indicate that all four options (ESPROFILE, LSPROFILE, RCPROFILE, AVPROFILE) are to be in effect. For details about variable names and the interpretation of the values in this data set, see the section “RESOURCEOUT= Usage Data Set” on page 127.

RESOURCESCHED=*SAS-data-set*

RESSCHED=*SAS-data-set*

RSCHEDULE=*SAS-data-set*

RSCHED=*SAS-data-set*

names the SAS data set in which you can save the schedules for each resource used by any activity. This option is valid whenever the RESOURCE statement is used to specify any resource requirements. The resulting data set is especially useful when resource-driven durations or resource calendars cause the resources used by an activity to have different schedules.

SETFINISHMILESTONE

specifies that milestones (zero duration activities) should have the same start and finish times as the finish time of their predecessor. In other words, this option enables milestones that mark the *end* of the preceding activity to coincide with its finish time. By default, if a milestone M is a successor to an activity that finishes at the end of the day (say 15Mar2004), the start and finish times for the milestone are specified as the beginning of the next day (16Mar2004). This corresponds to the definition of start times in the CPM procedure: *all* start times indicate the *beginning* of the date specified. For zero duration activities, the finish time is defined to be the same as the start time. The SETFINISHMILESTONE option specifies that the start and finish times for the milestone M should be specified as 15Mar2004, with the interpretation that the milestone’s schedule corresponds to the *end* of the day. There may be exceptions to this definition if there are special alignment constraints on the milestone. For details, see the section “Finish Milestones” on page 98.

SUPPRESSOBSWARN

turns off the display of warnings and notes for every observation with invalid or missing specifications.

WORKDATA=*SAS-data-set*

WORKDAY=*SAS-data-set*

identifies a SAS data set that defines the work pattern during a standard working day. Each numeric variable in this data set (also referred to as the **Workday** data set) is assumed to denote a unique shift pattern during one working day. The variables must be formatted as SAS time values and the observations are assumed to specify, alternately, the times when consecutive shifts start and end. See the section "Multiple Calendars" on page 101 for a description of this data set.

XFERVARS

indicates that all relevant variables are to be copied from the Activity data set to the Schedule data set. This includes all variables used in the ACTUAL statement, the ALIGNDATE and ALIGNTYPE statements, the SUCCESSOR statement, and the RESOURCE statement.

ACTIVITY Statement

ACTIVITY *variable* **;**

ACT *variable* **;**

The ACTIVITY statement is required when data are input in an AON format; this statement identifies the variable that contains the names of the nodes in the network. The activity associated with each node has a duration equal to the value of the DURATION variable. The ACTIVITY variable can be character or numeric because it is treated symbolically. Each node in the network must be uniquely defined.

The ACTIVITY statement is also supported in the Activity-on-Arc format. The ACTIVITY variable is used to uniquely identify the activity specified between two nodes of the network. In the AOA format, if the ACTIVITY statement is not specified, each observation in the Activity data set is treated as a new activity.

ACTUAL Statement

ACTUAL */ actual options* **;**

The ACTUAL statement identifies variables in the Activity data set that contain progress information about the activities in the project. For a project that is already in progress, you can describe the actual status of any activity by specifying the activity's actual start, actual finish, remaining duration, or percent of work completed. At least one of the four variables (A_START, A_FINISH, REMDUR, PCTCOMP) needs to be specified in the ACTUAL statement. These variables are referred to as *progress variables*. The TIMENOW= option in this statement represents the value of the current time (referred to as TIMENOW), and it is used in conjunction with the values of the progress variables to check for consistency and to determine default values if necessary.

You can also specify options in the ACTUAL statement that control the updating of the project schedule. Using the ACTUAL statement causes four new variables (A_START, A_FINISH, A_DUR, and STATUS) to be added to the Schedule data set; these variables are defined in the section "OUT= Schedule Data Set" on page 99. See the section "Progress Updating" on page 109 for more information.

The following options can be specified in the ACTUAL statement after a slash (/).

A_FINISH=*variable*

AF=*variable*

identifies a variable in the Activity data set that specifies the actual finish times of activities that are already completed. The actual finish time of an activity must be less than TIMENOW.

A_START=*variable*

AS=*variable*

identifies a variable in the Activity data set that specifies the actual start times of activities that are in progress or that are already completed. The actual start time of an activity must be less than TIMENOW.

AUTOUPDT

requests that PROC CPM should assume automatic completion (or start) of activities that are predecessors to activities already completed (or in progress). For example, if activity B is a successor of activity A, and B has an actual start time (or actual finish time or both) specified, while A has missing values for both actual start and actual finish times, then the AUTOUPDT option causes PROC CPM to assume that A must have already finished. PROC CPM then assigns activity A an actual start time and an actual finish time consistent with the precedence constraints. The AUTOUPDT option is the default.

ESTIMATEPCTC

ESTPCTC

ESTPCTCOMP

ESTPROG

indicates that a variable named PCT_COMP is to be added to the Schedule output data set (and the Resource Schedule output data set) that contains the percent completion time for each activity (for each resource used by each activity) in the project. This value is 0 for activities that have not yet started and 100 for completed activities; for activities in progress, this value is computed using the actual start time, the value of TIMENOW, and the revised duration of the activity.

FIXASTART

specifies that the actual start time of an activity should not be overwritten if it is specified to be on a non-work day. By default, none of the start or finish times of an activity can occur during a non-work period corresponding to the activity's calendar. If the actual start time is specified on a non-work day, it is moved to the nearest work day. The FIXASTART option specifies that the actual start and finish times be left unchanged even if they coincide with a non-working time. Thus, if the actual start time is specified to be sometime on Sunday, it is left unchanged even if Sunday is a non-working day in the activity's calendar.

NOAUTOUPDT

requests that PROC CPM should not assume automatic completion of activities. (The NOAUTOUPDT option is the reverse of the AUTOUPDT option.) In other words, only those activities that have nonmissing actual start or nonmissing actual finish times or both (either specified as values for the A_START and A_FINISH variables or computed on the basis of the REMDUR or PCTCOMP variables and TIMENOW) are assumed to have started; all other activities have an implicit start time that is greater than or equal to TIMENOW. This option requires you to enter the progress information for all the activities that have started or are complete; an activity is assumed to be *pending* until one of the progress variables indicates that it has started.

PCTCOMP=*variable*

PCTCOMPLETE=*variable*

PCOMP=*variable*

identifies a variable in the Activity data set that specifies the percentage of the work that has been completed for the current activity. The values for this variable must be between 0 and 100. A value of 0 for this variable means that the current activity has not yet started. A value of 100 means that the activity is already complete. Once again, the value of the TIMENOW= option is used as a reference point to resolve the values specified for the PCTCOMP variable. See the section "Progress Updating" on page 109 for more information.

REMDUR=*variable*

RDURATION=*variable*

RDUR=*variable*

identifies a variable in the Activity data set that specifies the remaining duration of activities that are in progress. The values of this variable must be nonnegative: a value of 0 for this variable means that the activity in that observation is completed, while a value greater than 0 means that the activity is not yet complete (the remaining duration is used to revise the estimate of the original duration). The value of the TIMENOW parameter is used to determine an actual start time or an actual finish time or both for activities based on the value of the remaining duration. See the section "Progress Updating" on page 109 for further information.

SHOWFLOAT

This option in the ACTUAL statement indicates that PROC CPM should allow activities that are completed or in progress to have nonzero float. By default, all activities that are completed or in progress have the late start schedule set to be equal to the early start schedule and thus have both total float and free float equal to 0. If the SHOWFLOAT option is specified, the late start schedule is computed for in-progress and completed activities using the precedence and time constraints during the backward pass.

TIMENOW=*timenow*

CURRDATE=*timenow*

specifies the SAS date, time, or datetime value that is used as a reference point to resolve the values of the remaining duration and percent completion times when the ACTUAL statement is used. It can be thought of as the instant at the *beginning of the specified date*, when a *snapshot* of the project is taken; the actual start times or finish times or both are specified for all activities that have started or have been completed by the *end of the previous day*. If an ACTUAL statement is used without specification of the TIMENOW= option, the default value is set to be the time period following the maximum of all the actual start and finish times that have been specified; if there are no actual start or finish times, then TIMENOW is set to be equal to the current date. See the section "Progress Updating" on page 109 for further information regarding the TIMENOW= option and the ACTUAL statement.

TIMENOWSPLT

indicates that activities that are in progress at TIMENOW can be split at TIMENOW if they cause resource infeasibilities. During resource allocation, any activities with values of E_START less than TIMENOW are scheduled even if there are not enough resources (a warning message is printed to the log if this is the case). This is true even for activities that are in progress. The TIMENOWSPLT option permits an activity to be split into two segments at TIMENOW, allowing the second segment of the activity to be scheduled later when resource levels permit. See the section "Activity Splitting" on page 122 for information regarding activity segments. Activities with an alignment type of MS or MF are not allowed to be split; also, activities without resource requirements will not be split.

ALIGNDATE Statement

ALIGNDATE *variable* **;**

DATE *variable* **;**

ADATE *variable* **;**

The ALIGNDATE statement identifies the variable in the Activity data set that specifies the dates to be used to constrain each activity to start or finish on a particular date. The ALIGNDATE statement is used in conjunction with the ALIGNTYPE statement, which specifies the type of alignment. A missing value for the variables specified in the ALIGNDATE statement indicates that the particular activity has no restriction imposed on it.

PROC CPM requires that if the ALIGNDATE statement is used, then all start activities (activities with no predecessors) have nonmissing values for the ALIGNDATE variable. If any start activity has a missing ALIGNDATE value, it is assumed to start on the date specified in the PROC CPM statement (if such a date is given) or, if no date is given, on the earliest specified start date of all start activities. If none of the start activities has a start date specified and a project start date is not specified in the PROC CPM statement, the procedure stops execution and returns an error message. See the section "Time-Constrained Scheduling" on page 96 for information about how the variables specified in the ALIGNDATE and ALIGNTYPE statements affect the schedule of the project.

ALIGNTYPE Statement

ALIGNTYPE *variable* **;**

ALIGN *variable* **;**

ATYPE *variable* **;**

The ALIGNTYPE statement is used to specify whether the date value in the ALIGNDATE statement is the earliest start date, the latest finish date, and so forth, for the activity in the observation. The values allowed for the variable specified in the ALIGNTYPE statement are specified in Table 4.2.

Table 4.2 Valid Values for the ALIGNTYPE Variable

Value	Type of Alignment
SEQ	Start equal to
SGE	Start greater than or equal to
SLE	Start less than or equal to
FEQ	Finish equal to
FGE	Finish greater than or equal to
FLE	Finish less than or equal to
MS	Mandatory start equal to
MF	Mandatory finish equal to

If an ALIGNDATE statement is specified without an ALIGNTYPE statement, all of the activities are assumed to have an aligntype of SGE. If an activity has a nonmissing value for the ALIGNDATE variable and a missing value for the ALIGNTYPE variable, then the aligntype is assumed to be SGE. See the section "Time-Constrained Scheduling" on page 96 for information about how the ALIGNDATE and ALIGNTYPE variables affect project scheduling.

BASELINE Statement

BASELINE / *options* **;**

The BASELINE statement enables you to save a specific schedule as a *baseline* or *target* schedule and compare another schedule, such as an updated schedule or resource constrained schedule, against it. The schedule that is to be saved as a baseline can be specified either by explicitly identifying two numeric variables in the input data set as the B_START and B_FINISH variables, or by indicating the particular schedule (EARLY, LATE, ACTUAL, or RESOURCE constrained schedule) that is to be used to set the B_START and B_FINISH variables. The second method of setting the schedule is useful when you want to set the baseline schedule on the basis of the *current invocation* of PROC CPM.

Note that the BASELINE statement needs to be specified in order for the baseline start and finish times to be copied to the Schedule data set. Just including the B_START and B_FINISH variables in the Activity data set does not initiate baseline processing.

The following options can be specified in the BASELINE statement after a slash (/).

B_FINISH=*variable*

BF=*variable*

specifies the numeric-valued variable in the Activity data set that sets B_FINISH.

B_START=*variable*

BS=*variable*

specifies the numeric-valued variable in the Activity data set that sets B_START.

COMPARE=*schedule*

compares a specific schedule (EARLY, LATE, RESOURCE or ACTUAL) in the Activity data set with the baseline schedule. The COMPARE option is valid only if the input data set already has a B_START and a B_FINISH variable or if the SET= option is also specified. In other words, the COMPARE option is valid only if there is a baseline schedule to compare with. The comparison is specified in two variables in the Schedule data set, S_VAR and F_VAR, which have the following definition:

```
S_VAR = Compare Start - B_START;
F_VAR = Compare Finish - B_FINISH;
```

where `Compare Start` and `Compare Finish` refer to the start and finish times corresponding to the schedule that is used as a comparison.

The values of the variables S_VAR and F_VAR are calculated in units of the INTERVAL= parameter, taking into account the calendar defined for the activity.

SET=*schedule*

specifies which of the four schedules (EARLY, LATE, RESOURCE, or ACTUAL) to set the baseline schedule equal to. The SET= option causes the addition of two new variables in the Schedule data set; these are the B_START and B_FINISH variables. The procedure sets B_START and B_FINISH equal to the start and finish times corresponding to the EARLY, LATE, ACTUAL, or RESOURCE schedules. If the Activity data set already has a B_START and B_FINISH variable, it is overwritten by the SET= option and a warning is displayed. The value RESOURCE is valid only if resource-constrained scheduling is being performed, and the value ACTUAL is valid only if the ACTUAL statement is present.

NOTE: The values ACTUAL, RESOURCE, and so on cause the B_START and B_FINISH values to be set to the *computed* values of A_START, S_START, . . . , and so on. They cannot be used to set the B_START and B_FINISH values to be equal to, say, A_START and A_FINISH or S_START and S_FINISH, if these variables are present in the Activity data set; to do that you must use B_START=A_START, B_FINISH=A_FINISH, and so on.

UPDATE=*schedule*

specifies the name of the schedule (EARLY, LATE, ACTUAL, or RESOURCE) that can be used to *update* the B_START and B_FINISH variables. This sets B_START and B_FINISH on the basis of the specified schedules *only* when the values of the baseline variables are missing in the Activity data set. The UPDATE option is valid only if the Activity data set already has B_START and B_FINISH. Note that if both the UPDATE= and SET= options are specified, the SET= specification is used.

CALID Statement

CALID *variable* **;**

The CALID statement specifies the name of a SAS variable that is used in the Activity, Holiday, and Calendar data sets to identify the calendar to which each observation refers. This variable can be either numeric or character depending on whether the different calendars are identified by unique numbers or names. If this variable is not found in any of the three data sets, PROC CPM looks for a default variable named _CAL_ in each data set (a warning message is then printed to the log). In the Activity data set, this variable specifies the calendar used by the activity in the given observation. Each calendar in the project is defined using the Workday, Calendar, and Holiday data sets. Each observation of the Calendar data set defines a standard work week through the shift patterns as defined by the Workday data set and a standard day length; these values are associated with the calendar identified by the value of the calendar variable in that observation. Likewise, each observation of the Holiday data set defines a holiday for the calendar identified by the value of the calendar variable.

If there is no calendar variable in the Activity data set, all activities are assumed to follow the default calendar. If there is no calendar variable in the Holiday data set, all of the holidays specified are assumed to occur in all the calendars. If there is no calendar variable in the Calendar data set, the first observation is assumed to define the default work week (which is also followed by any calendar that might be defined in the Holiday data set), and all subsequent observations are ignored. See the section “Multiple Calendars” on page 101 for further information.

DURATION Statement

DURATION *variable / options* **;**

DUR *variable* **;**

The DURATION statement identifies the variable in the Activity data set that contains the length of time necessary to complete the activity. If the network is input in AOA format, then the variable identifies the duration of the activity denoted by the arc joining the TAILNODE and the HEADNODE. If the network is input in AON format, then the variable identifies the duration of the activity specified in the ACTIVITY statement. The variable specified must be numeric. The DURATION statement must be specified. The values of the DURATION variable are assumed to be in *interval* units, where *interval* is the value of the INTERVAL= option.

If you want the procedure to compute the durations of the activities based on specified start and finish times, you can specify the start and finish times in the Activity data set, identified by the variables specified in the START= and FINISH= options. By default, the computed duration is used only if the value of the DURATION variable is missing for that activity. The duration is computed in units of the INTERVAL= parameter, taking into account the calendar defined for the activity.

In addition to specifying a fixed duration for an activity, you can specify the amount of work required (in units of the INTERVAL parameter) from each resource for a given activity. The WORK variable enables you to specify resource-driven durations for an activity; these (possibly different) durations are used to calculate the length of time required for the activity to be completed.

The following options can be specified in the DURATION statement after a slash (/).

FINISH=*variable*
: specifies a variable in the Activity data set that is to be used in conjunction with the START variable to determine the activity's duration.

START=*variable*
: specifies a variable in the Activity data set that is to be used in conjunction with the FINISH variable to determine the activity's duration.

OVERRIDEDUR
: specifies that if the START= and FINISH= values are not missing, the duration computed from these values is to be used in place of the duration specified for the activity. In other words, the computed duration is used in place of the duration specified for the activity.

HEADNODE Statement

HEADNODE *variable* **;**

HEAD *variable* **;**

TO *variable* **;**

The HEADNODE statement is required when data are input in AOA format. This statement specifies the variable in the Activity data set that contains the name of the node on the head of an arrow in the project network. This node is identified with the event that signals the end of an activity on that arc. The variable specified can be either a numeric or character variable because the procedure treats this variable symbolically. Each node must be uniquely defined.

HOLIDAY Statement

HOLIDAY *variable / options* **;**

HOLIDAYS *variable / options* **;**

The HOLIDAY statement specifies the names of variables used to describe non-workdays in the Holiday data set. PROC CPM accounts for holidays only when the INTERVAL= option has one of the following values: DAY, WORKDAY, WEEKDAY, DTDAY, DTWRKDAY, DTHOUR, DTMINUTE, or DTSECOND. The HOLIDAY statement must be used with the HOLIDATA= option in the PROC CPM statement. Recall that the HOLIDATA= option identifies the SAS data set that contains a list of the holidays and non-workdays around which you schedule your project. Holidays are defined by specifying the start of the holiday (the HOLIDAY variable) and either the length of the holiday (the HOLIDUR variable) or the finish time of the holiday (the HOLIFIN variable). The HOLIDAY variable is mandatory with the HOLIDAY statement; the HOLIDUR and HOLIFIN variables are optional.

The HOLIDAY and HOLIFIN variables must be formatted as SAS date or datetime variables. If no format is associated with a HOLIDAY variable, it is assumed to be formatted as a SAS date value. If the schedule of the project is computed as datetime values (which is the case if INTERVAL is DTDAY, WORKDAY, and so on), the holiday variables are interpreted as follows:

- If the HOLIDAY variable is formatted as a date value, then the holiday is assumed to start at the value of the DAYSTART= option on the day specified in the observation and to end *d* units of *interval* later (where *d* is the value of the HOLIDUR variable and *interval* is the value of the INTERVAL= option).
- If the HOLIDAY variable is formatted as a datetime value, then the holiday is assumed to start at the date and time specified and to end *d* units of *interval* later.

The HOLIDUR and HOLIFIN variables are specified using the following options in the HOLIDAY statement:

HOLIDUR=*variable*

HDURATION=*variable*

identifies a variable in the Holiday data set that specifies the duration of the holiday. The INTERVAL= option specified on the PROC CPM statement is used to interpret the value of the holiday duration variables. Thus, if the duration of a holiday is specified as 2 and the value of the INTERVAL= option is WEEKDAY, the length of the holiday is interpreted as two weekdays.

HOLIFIN=*variable*

HOLIEND=*variable*

identifies a variable in the Holiday data set that specifies the finish time of the holiday defined in that observation. If a particular observation contains both the duration as well as the finish time of the holiday, only the finish time is used; the duration is ignored.

ID Statement

ID *variables* **;**

The ID statement identifies variables not specified in the TAILNODE, HEADNODE, ACTIVITY, SUCCESSOR, or DURATION statements that are to be included in the Schedule data set. This statement is useful for carrying any relevant information about each activity from the Activity data set to the Schedule data set.

PROJECT Statement

PROJECT *variable / options* **;**

PARENT *variables / options* **;**

The PROJECT statement specifies the variable in the Activity data set that identifies the project to which an activity belongs. This variable must be of the same type and length as the variable defined in the ACTIVITY statement. A project can also be treated as an activity with precedence and time constraints. In other words, any value of the PROJECT variable can appear as a value of the ACTIVITY variable, and it can have specifications for the DURATION, ALIGNDATE, ALIGNTYPE, ACTUAL, RESOURCE, and SUCCESSOR variables. However, some of the interpretations of these variables for a project (or supertask) may be different from the corresponding interpretation for an activity at the lowest level. See the section "Multiproject Scheduling" on page 131 for an explanation.

The following options can be specified in the PROJECT statement after a slash (/).

AGGREGATEPARENTRES

AGGREGATEP_RES

AGGREGPR

indicates that the resource requirements for all supertasks are to be used only for aggregation purposes and not for resource-constrained scheduling.

DESCENDING

DESC

indicates that, in addition to the ascending sort variables (ES_ASC, LS_ASC, and SS_ASC) that are requested by the ESORDER, LSORDER, and SSORDER options, the corresponding descending sort variables (ES_DESC, LS_DESC, and SS_DESC, respectively) are also to be added to the Schedule output data set.

ESORDER

ESO

indicates that a variable named ES_ASC is to be added to the Schedule output data set; this variable can be used to order the activities in such a way that the activities within each subproject are in increasing order of the early start time. This order is not necessarily the same as the one that would be obtained by sorting all the activities in the Schedule data set by E_START.

IGNOREPARENTRES

IGNOREP_RES

IGNOREPR

indicates that the resource requirements for all supertasks are to be ignored.

LSORDER

LSO

indicates that a variable named LS_ASC is to be added to the Schedule output data set; this variable can be used to order the activities in such a way that the activities within each subproject are in increasing order of the late start time.

ORDERALL

ALL

is equivalent to specifying the ESORDER and LSORDER options (and the SSORDER option when resource constrained scheduling is performed).

RSCHEDORDER

RSCHDORD

RSORDER

indicates that the order variables that are included in the Schedule output data set are also to be included in the Resource Schedule output data set.

RSCHEDWBS

RSCHDWBS

RSWBS

indicates that the WBS code is also to be included in the Resource Schedule data set.

SEPCRIT

computes individual critical paths for each project. By default, the master project's early finish time is treated as the starting point for the calculation of the backward pass (which calculates the late start schedule). The late finish time for each subproject is then determined during the backward pass on the basis of the precedence constraints. If a time constraint is placed on the finish time of a subproject (using the ALIGNDATE and ALIGNTYPE variables), the late finish time of the subproject is further constrained by this value.

The SEPCRIT option, on the other hand, requires the late finish time of each subproject to be less than or equal to the early finish time of the subproject. Thus, if you have a set of independent, parallel projects, the SEPCRIT option enables you to compute separate critical paths for each of the subprojects.

SSORDER

SSO

indicates that a variable named SS_ASC is to be added to the Schedule output data set; this variable can be used to order the activities in such a way that the activities within each subproject are in increasing order of the resource-constrained start time.

USEPROJDUR

USEPROJDURSPEC

USESPECDUR

uses the specified subproject duration to compute the maximum allowed late finish for each subproject. This is similar to the SEPCRIT option, except that the *specified project duration* is used to set an upper bound on each subproject's late finish time instead of the *project span* as computed from the span of all the subtasks of the project. In other words, if E_START and E_FINISH are the early start and finish times of the subproject under consideration, and the subproject duration is PROJ_DUR, where

PROJ_DUR = E_FINISH - E_START

then the SEPCRIT option sets

L_FINISH ≤ E_START + PROJ_DUR

while the USEPROJDUR option sets

L_FINISH ≤ E_START + DUR

where DUR is the duration specified for the subproject in the Activity data set.

WBSCODE

WBS

ADDWBS

indicates that the CPM procedure is to compute a WBS code for the activities in the project using the project hierarchy structure specified. This code is computed for each activity and stored in the variable WBS_CODE in the Schedule output data set.

RESOURCE Statement

RESOURCE *variables / resource options* **;**

RES *variables / resource options* **;**

The RESOURCE statement identifies the variables in the Activity data set that contain the levels of the various resources required by the different activities. This statement is necessary if the procedure is required to summarize resource utilization for various resources.

This statement is also required when the activities in the network use limited resources and a schedule is to be determined subject to resource constraints in addition to precedence constraints. The levels of the various resources available are obtained from the RESOURCEIN= data set (the Resource data set.) This data set need not contain all of the variables listed in the RESOURCE statement. If any resource variable specified in the RESOURCE statement is not also found in the Resource data set, it is assumed to be available in unlimited quantity and is not used in determining the constrained schedule.

The following options are available with the RESOURCE statement to help control scheduling the activities subject to resource constraints. Some control the scheduling heuristics, some control the amount of information to be output to the RESOURCEOUT= data set (the Usage data set), and so on.

ACTDELAY=*variable*

specifies the name of a variable in the Activity data set that specifies a value for the maximum amount of delay allowed for each activity. The values of this variable should be greater than or equal to 0. If a value is missing, the value of the DELAY= option is used instead.

ACTIVITYPRTY=*variable*

ACTPRTY=*variable*

identifies the variable in the Activity data set that contains the priority of each activity. This option is required if resource-constrained scheduling is to be performed and the scheduling rule specified is ACTPRTY. If the value of the SCHEDRULE= option is specified as the keyword ACTPRTY, then all activities waiting for resources are ordered by increasing values of the ACTPRTY= variable. Missing values of the activity priority variable are treated as +INFINITY. See the section "Scheduling Method" on page 119 for a description of the various scheduling rules used during resource constrained scheduling.

ADDCAL

requests that a variable, _CAL_, be added to the Resource Schedule data set that identifies the resource calendar for each resource used by each activity. For observations that summarize the activity's schedule, this variable identifies the activity's calendar.

ALL

is equivalent to specifying the ESPROFILE and LSPROFILE options when an unconstrained schedule is obtained and is equivalent to specifying all four options, AVPROFILE (AVP), ESPROFILE (ESP), LSPROFILE (LSP), and RCPROFILE (RCP), when a resource-constrained schedule is obtained. If none of these four options are specified and a Usage data set is specified, by default the ALL option is assumed to be in effect.

ALTBEFORESUP

indicates that all alternate resources are to be checked first before using supplementary resources. By default, if supplementary levels of resources are available, the procedure uses supplementary levels first and uses alternate resources only if the supplementary levels are not sufficient.

APPEND

APPENDINTXRATE

APPENDRATEXINT

APPENDUSAGE

indicates that the Usage data set is to contain two sets of observations: the first set indicates the *rate* of usage for each resource at the beginning of the current time period, and the second set contains the *total* usage of each resource for the current time period. In other words, the Usage data set appends observations indicating the total usage of each resource to the default set of observations. If the APPEND option is specified, the procedure adds a variable named OBS_TYPE to the Usage data set. This variable contains the value 'RES_RATE' for the observations that indicate rate of usage and the value 'RES_USED' for the observations that indicate the total usage.

AROUTCAL=*calname*

specifies the name of the calendar to be used for incrementing the _TIME_ variable in the Usage data set.

AVPROFILE

AVP

AVL

creates one variable in the Usage data set corresponding to each variable in the RESOURCE statement. These new variables denote the amount of resources remaining after resource allocation. This option is ignored if resource allocation is not performed.

AWAITDELAY

forces PROC CPM to wait until L_START+*delay*, where *delay* is the maximum delay allowed for the activity (which is the value of the ACTDELAY= variable or the DELAY= option), before an activity is scheduled using supplementary levels of resources. By default, even if an activity has a nonzero value specified for the ACTDELAY= variable (or the DELAY= option), it may be scheduled using supplementary resources before L_START+*delay*. This happens if the procedure does not see any increase in the resource availability in the future. Thus, if it appears that the activity will require supplementary resources anyway, the procedure may schedule it before L_START+*delay*. The AWAITDELAY option prohibits this behavior; it will not use supplementary resources to schedule an activity before L_START+*delay*. This option can be used to force activities with insufficient resources to start at L_START by setting DELAY=0.

CUMUSAGE

specifies that the Usage data set should indicate the cumulative usage of consumable resources. Note that by default, for consumable resources, each observation in the Usage data set contains the rate of usage for each resource at the start of the given time interval. See the section "RESOURCEOUT= Usage Data Set" on page 127 for a definition of the variables in the resource usage output data set. In some applications, it may be useful to obtain the cumulative usage of these resources. The CUMUSAGE option can be used to obtain the cumulative usage of consumable resources up to the time specified in the _TIME_ variable.

DELAY=*delay*

specifies the maximum amount by which an activity can be delayed due to lack of resources. If E_START of an activity is 1JUN04 and L_START is 5JUN04 and *delay* is specified as 2, PROC CPM first tries to schedule the activity to start on June 1, 2004. If there are not enough resources to schedule the activity, the CPM procedure postpones the activity's start time. However, it does not postpone the activity beyond June 7, 2004 (because *delay*=2 and L_START=5JUN04).

If the activity cannot be scheduled even on 7JUN04, then PROC CPM tries to schedule it by using supplementary levels of resources, if available, or by using alternate resources, if possible. If resources are still not sufficient, the procedure stops with an error message. The default value of the DELAY= option is assumed to be +INFINITY.

DELAYANALYSIS

SLIPINF

causes the addition of three new variables to the Schedule data set. The variables are R_DELAY, DELAY_R and SUPPL_R. The R_DELAY variable indicates the number of units (in *interval* units) by which the activity's schedule has slipped due to resource unavailability, and the DELAY_R variable contains the name of the resource, the *delaying resource*, that has caused the slippage.

The R_DELAY variable is calculated as follows: it is the difference between S_START and the time when an activity first enters the list of activities that are available to be scheduled. (See the section

"Scheduling Method" on page 119 for a definition of this waiting list of activities.) R_DELAY is not necessarily the same as S_START − E_START.

If several resources are insufficient, causing a delay in the activity, DELAY_R is the name of the resource that *first* causes an activity to be postponed.

The variable SUPPL_R contains the name of the *first* resource that is used above the primary level in order for an activity to be scheduled at S_START.

ESPROFILE

ESP

ESS

creates one variable in the Usage data set corresponding to each variable in the RESOURCE statement. Each new variable denotes the resource usage based on the early start schedule for the corresponding resource variable.

E_START

requests that the E_START and E_FINISH variables, namely the variables specifying the early start schedule, be included in the Schedule data set in addition to the S_START and S_FINISH variables. This option is the default and can be turned off using the NOE_START option.

EXCLUNSCHED

excludes the resource consumption corresponding to unscheduled activities from the daily resource usage reported for each time period in the Usage data set. The Usage data set contains a variable named R*resname* for each resource variable *resname*. For each observation in this data set, each such variable contains the total amount of resource (*rate of usage* for a consumable resource) used by all the activities that are active at the time period corresponding to that observation. By default, this calculation includes even activities that are still unscheduled when resource constrained scheduling is stopped either by the STOPDATE= option or due to resource infeasibilities. The EXCLUNSCHED option enables the exclusion of activities that are still unscheduled. The unscheduled activities are assumed to start as per the early start schedule (unless the UPDTUNSCHED option is specified).

FILLUNSCHED

FILLMISSING

fills in S_START and S_FINISH values for activities that are still unscheduled when resource constrained scheduling is stopped either by the STOPDATE= option or due to resource infeasibilities. By default, the Schedule data set contains missing values for S_START and S_FINISH corresponding to unscheduled activities. If the FILLUNSCHED option is on, the procedure uses the original E_START and E_FINISH times for these activities. If the UPDTUNSCHED option is also specified, the procedure uses *updated* values.

F_FLOAT

requests that the Schedule data set include the F_FLOAT variable computed using the unconstrained early and late start schedules. If resource allocation is not performed, this variable is always included in the output data set.

INCLUNSCHED

enables the inclusion of activities that are still unscheduled in the computation of daily (or cumulative) resource usage in the Usage data set when resource-constrained scheduling is stopped either by the STOPDATE= option or due to resource infeasibilities. This option is the default and can be turned off by the EXCLUNSCHED option.

INDEPENDENTALLOC

INDEPALLOC

enables each resource to be scheduled independently for each activity during resource-constrained scheduling. Consider the basic resource scheduling algorithm described in the section "Scheduling Method" on page 119. When all the precedence requirements of an activity are satisfied, the activity is inserted into the list of activities that are waiting for resources using the appropriate scheduling rule. An activity in this list is scheduled to start at a particular time only if *all* the resources required by it are available in sufficient quantity. Even if the resources are required by the activity for different lengths of time, or if the resources have different calendars, all resources must be available to start at that particular time (or at the beginning of the next work period for the resource's calendar).

If you specify the INDEPENDENTALLOC option, however, each resource is scheduled independently of the others. This may cause an activity's schedule to be extended if its resources cannot all start at the same time.

INFEASDIAGNOSTIC

INFEASDIAG

requests PROC CPM to continue scheduling even when resources are insufficient. When PROC CPM schedules the project subject to resource constraints, the scheduling process is stopped when the procedure cannot find sufficient resources for an activity before the activity's latest possible start time (accounting for the DELAY= or ACTDELAY= options and using supplementary or alternate resources if necessary and if allowed). The INFEASDIAGNOSTIC option can be used to override this default action. (Sometimes, you may want to know the level of resources needed to schedule a project to completion even if resources are insufficient.) This option is equivalent to specifying infinite supplementary levels for all the resources under consideration; the DELAY= value is assumed to equal the default value of +INFINITY, unless otherwise specified.

LSPROFILE

LSP

LSS

creates one variable in the Usage data set corresponding to each variable in the RESOURCE statement. Each new variable denotes the resource usage based on the late start schedule for the corresponding resource variable.

L_START

requests that the L_START and L_FINISH variables, namely the variables specifying the late start schedule, be included in the Schedule data set in addition to the S_START and S_FINISH variables. This option is the default and can be turned off using the NOL_START option.

MAXDATE=*maxdate*

specifies the maximum value of the _TIME_ variable in the Usage data set. The default value of *maxdate* is the maximum finish time for all of the schedules for which a usage profile was requested.

MAXNSEGMT=*variable*

MAXNSEG=*variable*

specifies a variable in the Activity data set that indicates the maximum number of segments that the current activity can be split into. A missing value for this variable is set to a default value that depends on the duration of the activity and the value of the MINSEGMTDUR variable. A value of 1 indicates that the activity cannot be split. By default, PROC CPM assumes that any activity, once started, cannot be stopped until it is completed (except for breaks due to holidays or weekends). Thus, even during resource-constrained scheduling, an activity is scheduled only if enough resources can be found for it throughout its *entire* duration. Sometimes, you may want to allow preemption of activities already in progress; thus, a more *critical* activity could cause another activity to be split into two or more segments.

However, you may not want a particular activity to be split into too many segments, or to be split too many times. The MAXNSEGMT= and MINSEGMTDUR= options enable you to control the number of splits and the length of each segment.

MAXOBS=*max*

specifies an upper limit on the number of observations that the Usage data set can contain. If the values specified for the ROUTINTERVAL= and ROUTINTPER= options are such that the data set will contain more than *max* observations, then PROC CPM does not create the output data set and stops with an error message.

The MAXOBS= option is useful as a check to ensure that a very large data set (with several thousands of observations) is not created due to a wrong specification of the ROUTINTERVAL= option. For example, if *interval* is DTYEAR and *routinterval* is DTHOUR and the project extends over 2 years, the number of observations would exceed 15,000. The default value of the MAXOBS= option is 1000.

MILESTONERESOURCE

specifies that milestone activities consume resources. If a nonzero requirement is specified for a milestone, the corresponding consumable resources are used at the scheduled time of that milestone.

MILESTONENORESOURCE

specifies that milestone activities do not consume resources. This implies that all resource requirements are ignored for milestone activities. This is the default behavior.

MINDATE=*mindate*

specifies the minimum value of the _TIME_ variable in the Usage data set. The default value of *mindate* is the minimum start time for all of the schedules for which a usage profile is requested. Thus, the Usage data set has observations containing the resource usage and availability information from *mindate* through *maxdate* .

MINSEGMTDUR=*variable*

MINSEGD=*variable*

specifies a variable in the Activity data set that indicates the minimum duration of any segment of the current activity. A missing value for this variable is set to a value equal to one fifth of the activity's duration.

MULTIPLEALTERNATES

MULTALT

indicates that multiple alternate resources can be used to substitute for a single resource. In other words, if one of the alternate resources is not sufficient to substitute for the primary resource, the procedure will use other alternates, as needed, to fulfill the resource requirement. For example, if an activity needs 1.5 programmers and the allowed alternates are JOHN and MARY, the procedure will use JOHN (at rate 1) and MARY (at rate 0.5) to allocate a total of 1.5 programmers. See the section "Specifying Multiple Alternates" on page 125 for details.

NOE_START

requests that the E_START and E_FINISH variables, namely the variables specifying the early start schedule, be dropped from the Schedule data set. Note that the default is E_START. Also, if resource allocation is not performed, the NOE_START option is ignored.

NOF_FLOAT

requests that the F_FLOAT variable be dropped from the Schedule data set when resource-constrained scheduling is requested. This is the default behavior. To include the F_FLOAT variable in addition to the resource-constrained schedule, use the F_FLOAT option. If resource allocation is not performed, F_FLOAT is always included in the Schedule data set.

NOL_START

requests that the Schedule data set does not include the late start schedule, namely, the L_START and L_FINISH variables. Note that the default is L_START. Also, if resource allocation is not performed, the NOL_START option is ignored.

NORESOURCEVARS

NORESVARSOUT

NORESVARS

requests that the variables specified in the RESOURCE statement be dropped from the Schedule data set. By default, all of the resource variables specified on the RESOURCE statement are also included in the Schedule data set.

NOT_FLOAT

requests that the T_FLOAT variable be dropped from the Schedule data set when resource-constrained scheduling is requested. This is the default behavior. To include the T_FLOAT variable in addition to the resource-constrained schedule, use the T_FLOAT option. If resource allocation is not performed, T_FLOAT is always included in the Schedule data set.

NROUTCAL=*calnum*

specifies the number of the calendar to be used for incrementing the _TIME_ variable in the Usage data set.

OBSTYPE=*variable*

specifies a character variable in the Resource data set that contains the type identifier for each observation. Valid values for this variable are RESLEVEL, RESTYPE, RESUSAGE, RESPRTY, SUPLEVEL, ALTRATE, ALTPRTY, RESRCDUR, CALENDAR, MULTALT, MINARATE, and AUXRES. If OBSTYPE= is not specified, then all observations in the data set are assumed to denote the levels of the resources, and all resources are assumed to be replenishable and constraining.

PERIOD=*variable*

PER=*variable*

identifies the variable in the RESOURCEIN= data set that specifies the date from which a specified level of the resource is available for each observation with the OBSTYPE variable equal to 'RESLEVEL'. It is an error if the PERIOD= variable has a missing value for any observation specifying the levels of the resources or if the Resource data set is not sorted in increasing order of the PERIOD= variable.

RCPROFILE

RCP

RCS

creates one variable in the Usage data set corresponding to each variable in the RESOURCE statement. Each new variable denotes the resource usage based on the resource-constrained schedule for the corresponding resource variable. This option is ignored if resource allocation is not performed.

RESCALINTERSECT

RESCALINT

RCI

specifies that an activity can be scheduled only during periods that are common working times for all resource calendars (corresponding to the resources used by that activity) and the activity's calendar. This option is valid only if multiple calendars are in use and if calendars are associated with individual resources. Use this option with caution; if an activity uses resources that have mutually disjoint calendars, that activity can never be scheduled. For example, if one resource works a night shift while another resource works a day shift, the two calendars do not have any common working time.

Only primary resources are included in the intersection; any alternate or auxiliary resources are not included when determining the common working calendar for the activity.

If you do not specify the RESCALINTERSECT option, and resources have independent calendars, then the procedure schedules each resource using its own calendar. Thus, an activity can have one resource working on a five-day calendar, while another resource is working on a seven-day calendar.

RESID=*variable*

specifies a variable in the RESOURCEIN= data set that indicates the name of the resource variable for which *alternate resource information* or *auxiliary resource information* is being specified in that observation.

Observations that indicate alternate resources are identified by the values 'ALTRATE' and 'ALTPRTY' for the OBSTYPE variable. These values indicate whether the observation specifies a *rate of substitution* or a *priority for substitution*; the value of the RESID variable in such an observation indicates the particular resource for which alternate resource information is specified in that observation. The specification of the RESID= option triggers the use of alternate resources. See the section "Specifying Alternate Resources" on page 123 for further information.

Observations indicating auxiliary resources are identified by the value 'AUXRES' for the OBSTYPE variable. Such observations specify the name of the primary resource as the value of the RESID variable and the rate of auxiliary resources needed for every unit of the primary resource as values of the other resource variables. See the section "Auxiliary Resources" on page 127 for further information.

RESOURCEVARS

RESVARSOUT

requests that the variables specified in the RESOURCE statement be included in the Schedule data set. These include the RESOURCE variables identifying the resource requirements, the activity priority variable, the activity delay variable, and any variables specifying activity splitting information. This option is the default and can be turned off by the NORESVARSOUT option.

ROUTINTERVAL=*routinterval*

STEPINT=*routinterval*

specifies the units to be used to determine the time interval between two successive values of the _TIME_ variable in the Usage data set. It can be used in conjunction with the ROUTINTPER= option to control the amount of information to be included in the data set. Valid values for *routinterval* are DAY, WORKDAY, WEEK, MONTH, WEEKDAY, QTR, YEAR, DTDAY, DTWRKDAY, DTWEEK, DTMONTH, DTQTR, DTYEAR, DTSECOND, DTMINUTE, DTHOUR, SECOND, MINUTE, or HOUR. The value of this parameter must be chosen carefully; a massive amount of data could be generated by a bad choice. If this parameter is not specified, a default value is chosen depending on the format of the schedule variables.

ROUTINTPER=*routintper*

STEPSIZE=*routintper*

STEP=*routintper*

specifies the number of *routinterval* units between successive observations in the Usage data set where *routinterval* is the value of the ROUTINTERVAL= option. For example, if *routinterval* is MONTH and *routintper* is 2, the time interval between each pair of observations in the Usage data set is two months. The default value of *routintper* is 1. If *routinterval* is blank (‘ ’), then *routintper* can be used to specify the exact numeric interval between two successive values of the _TIME_ variable in the Usage data set. *routintper* is only allowed to have integer values when *routinterval* is specified as one of the following: WEEK, MONTH, QTR, YEAR, DTWEEK, DTMONTH, DTQTR, or DTYEAR.

ROUTNOBREAK

ROUTCONT

specifies that the _TIME_ variable is to be incremented using a calendar with no breaks or holidays. Thus, the Usage data set contains one observation per unit *routinterval* from *mindate* to *maxdate*, without any breaks for holidays or weekends. By default, the _TIME_ variable is incremented using the default calendar; thus, if the default calendar follows a five-day work week, the Usage data set skips weekends.

RSCHEDID=*(variables)*

RSID=*(variables)*

identifies variables not specified in the TAILNODE, HEADNODE, or ACTIVITY statements that are to be included in the Resource Schedule data set. This option is useful for carrying any relevant information about each activity from the Activity data set to the Resource Schedule data set.

SCHEDRULE=*schedrule*

RULE=*schedrule*

specifies the rule to be used to order the list of activities whose predecessor activities have been completed while scheduling activities subject to resource constraints. Valid values for *schedrule* are LST, LFT, SHORTDUR, ACTPRTY, RESPRTY, and DELAYLST. (See the section “Scheduling Rules” on page 120 for more information.) The default value of SCHEDRULE is LST. If an invalid specification is given for the SCHEDRULE= option, the default value is used, and a warning message is displayed in the log.

SCHEDRULE2=*schedrule2*

RULE2=*schedrule2*

specifies the rule to be used to break ties caused by the SCHEDRULE= option. Valid values for *schedrule2* are LST, LFT, SHORTDUR, ACTPRTY, RESPRTY, and DELAYLST. ACTPRTY and RESPRTY cannot be specified at the same time for the two scheduling rules; in other words, if *schedrule* is ACTPRTY, *schedrule2* cannot be RESPRTY and vice versa.

SETFINISH=MAX | EARLY (Experimental)

controls the computation of resource-constrained finish times for activities that are in progress. A value of EARLY sets the resource-constrained finish time to the early finish time as derived from the progress updating variables A_START, A_FINISH, REMDUR, and PCTCOMP. Specifying the default value of MAX sets the resource-constrained finish time to the maximum of the early finish time and the finish times for all resources for the given activity. Use of the EARLY value for this option could leave work unfulfilled because of the priority given to the progress updating information.

SPLITFLAG

indicates that activities are allowed to be split into segments during resource allocation. This option can be used instead of specifying either the MAXNSEGMT= or the MINSEGMTDUR= variable; PROC CPM assumes that the activity can be split into no more than five segments.

STOPDATE=*stdate*

specifies the cutoff date for resource-constrained scheduling. When such a date is specified, S_START and S_FINISH are set to missing beyond the cutoff date. Options are available to set these missing values to the original E_START and E_FINISH times (FILLUNSCHED) or to updated values based on the scheduled activities (UPDTUNSCHED).

T_FLOAT

requests that the Schedule data set include the T_FLOAT variable computed using the unconstrained early and late start schedules. Note that if resource allocation is not performed, this variable is always included in the Schedule data set.

TOTUSAGE

INTXRATE

INTUSAGE

RATEXINT

specifies that the Usage data set is to indicate the *total* usage of the resource for the current time period. The current time period is the time interval from the time specified in the _TIME_ variable for the current observation to the time specified in the _TIME_ variable for the next observation. The total usage is computed taking into account the relevant activity and resource calendars. Note that, by default, the observations in the Usage data set specify the *rate* of usage for each resource at the beginning of the current time period. The TOTUSAGE option specifies the *product* of the rate and the time interval between two successive observations. To get both the *rate* and the *product*, use the APPEND option.

UNSCHEDMISS

sets the S_START and S_FINISH values to missing for activities that are still unscheduled when resource constrained scheduling is stopped either by the STOPDATE= option or due to resource infeasibilities. This is the default and can be turned off by specifying the FILLUNSCHED option.

UPDTUNSCHED

causes the procedure to use the S_START and S_FINISH times of *scheduled* activities to update the *projected* start and finish times for the activities that are still unscheduled when resource constrained scheduling is stopped either by the STOPDATE= option or due to resource infeasibilities. These updated dates are used as the S_START and S_FINISH times.

WORK=*variable*

identifies a variable in the Activity data set that specifies the total amount of work required by one unit of a resource. This work is represented in units of the INTERVAL parameter. The procedure uses the rate specified for the resource variable to compute the duration of the activity for that resource. Thus, if the value of the WORK variable is 10, and the value of the resource variable R1 is 2, then the activity requires 5 *interval* units for the resource R1. For details, see the section “Resource-Driven Durations and Resource Calendars” on page 112.

SUCCESSOR Statement

SUCCESSOR *variables / lag options* **;**

SUCC *variables / lag options* **;**

The SUCCESSOR statement is required when data are input in an AON format. This statement specifies the variables that contain the names of the immediate successor nodes (activities) to the ACTIVITY node. These variables must be of the same type and length as those defined in the ACTIVITY statement.

If the project does not have any precedence relationships, it is not necessary to use the SUCCESSOR statement. Thus, you can specify only the ACTIVITY statement without an accompanying SUCCESSOR statement.

If the precedence constraints among the activities have some nonstandard relationships, you can specify these using the LAG options. The following is a list of LAG options.

ALAGCAL=*calname*

specifies the name of the calendar to be used for all lags. The default value for this option is the DEFAULT calendar.

LAG=*variables*

specifies the variables in the Activity data set used to identify the lag relationship (lag type, duration, and calendar) between the activity and its successor. The LAG variables must be character variables. You can specify as many LAG variables as there are SUCCESSOR variables; each SUCCESSOR variable is matched with the corresponding LAG variable. You must specify the LAG variables enclosed in parentheses. In a given observation, the *i*th LAG variable specifies the type of relation between the current activity (as specified by the ACTIVITY variable) and the activity specified by the *i*th SUCCESSOR variable. If there are more LAG variables than SUCCESSOR variables, the extra LAG variables are ignored; conversely, if there are fewer LAG variables, the extra SUCCESSOR variables are all assumed to indicate successors with a *standard* (finish-to-start) relationship.

In addition to the type of relation, you can also specify a lag duration and a lag calendar in the same variable. The relation_lag_calendar information is expected to be specified as

keyword _ duration _ calendar

where *keyword* is one of ‘ ’, FS, SS, SF, or FF, *duration* is a number specifying the duration of the lag (in *interval* units), and *calendar* is either a calendar name or number identifying the calendar followed by the lag duration. A missing value for the *keyword* is assumed to mean the same as FS, which is the standard relation of *finish-to-start*. The other three values, SS, SF, and FF, denote relations of the type *start-to-start*, *start-to-finish*, and *finish-to-finish*, respectively. If there are no LAG variables, all relationships are assumed to be of the type *finish-to-start* with no lag duration. Table 4.3 contains some examples of lag specifications.

Table 4.3 Lag Specifications

Activity	Successor	LAG	Interpretation
A	B	SS_3	Start to start lag of 3 units
A	B	_5.5	Finish to start lag of 5.5 units
A	B	FF_4	Finish to finish lag of 4 units
A	B	_SS	Invalid and ignored (with warning)
A		SS_3	Ignored
A	B	SS_3_1	Start to start lag of 3 units w.r.t. calendar 1

NLAGCAL=*calnum*
specifies the number of the calendar to be used for all lags. The default value for this option is the DEFAULT calendar.

TAILNODE Statement

TAILNODE *variable* **;**

TAIL *variable* **;**

FROM *variable* **;**

The TAILNODE statement is required when data are input in AOA (arrow notation) format. It specifies the variable that contains the name of each node on the tail of an arc in the project network. This node is identified with the event that signals the *start* of the activity on that arc. The variable specified can be either a numeric or character variable since the procedure treats this variable symbolically. Each node must be uniquely defined.

Details: CPM Procedure

This section provides a detailed outline of the use of the CPM procedure. The material is organized in subsections that describe different aspects of the procedure. They have been placed in increasing order of functionality. The first section describes how to use PROC CPM to schedule a project subject only to precedence constraints. The next two sections describe some of the features that enable you to control the units of duration and specify nonstandard precedence constraints. In the section "Time-Constrained Scheduling" on page 96, the statements needed to place time constraints on the activities are introduced. the section "Finish Milestones" on page 98 describes some options controlling the treatment of milestones.

The section "OUT= Schedule Data Set" on page 99 describes the format of the schedule output data set (the Schedule data set). The section "Multiple Calendars" on page 101 deals with calendar specifications for the different activities.

The section "Baseline and Target Schedules" on page 109 describes how you can save specific schedules as baseline or target schedules. The section "Progress Updating" on page 109 describes how to incorporate the actual start and finish times for a project that is already in progress. The section "Resource-Driven Durations and Resource Calendars" on page 112 describes how the WORK variable can be used to specify resource-driven durations and the effect of resource calendars on the activity schedules.

Next, the section "Resource Usage and Allocation" on page 113 pertains to resource usage and resource-constrained scheduling and describes how to specify information about the resources and the resource requirements for the activities. The scheduling algorithm is also described in this section and some advanced features such as alternate resources, auxiliary resources, negative resource requirements, and so on, are discussed under separate subsections.

The section "RESOURCEOUT= Usage Data Set" on page 127 describes the format of the resource usage output data set (the Usage data set) and explains how to interpret the variables in it.

When resource-driven durations are specified or resource calendars are in effect, each resource used by an activity may have a different schedule. In this case, the Resource Schedule data set, described in the section "RESOURCESCHED= Resource Schedule Data Set" on page 131, contains the individual resource schedules for each activity.

The section "Multiproject Scheduling" on page 131 describes how you can use PROC CPM when there are multiple projects that have been combined together in a multiproject structure.

PROC CPM also defines a macro variable that is described in the section "Macro Variable _ORCPM_" on page 134.

Table 4.9 in the section "Input Data Sets and Related Variables" on page 135 lists all the variables used by the CPM procedure and the data sets that contain them. Table 4.10 in the section "Missing Values in Input Data Sets" on page 137 lists all of the variables in the different input data sets and describes how PROC CPM treats missing values corresponding to each of them. Finally, the section "FORMAT Specification" on page 139 underlines the importance of associating the correct FORMAT specification with all the date-type variables, and the section "Computer Resource Requirements" on page 139 indicates the storage and time requirements of the CPM procedure.

Scheduling Subject to Precedence Constraints

The basic function of the CPM procedure is to determine a schedule of the activities in a project subject to precedence constraints among them. The minimum amount of information that is required for a successful invocation of PROC CPM is the network information specified either in AON or AOA formats and the duration of each activity in the network. The INTERVAL= option specifies the units of duration, and the DATE= option specifies a start date for the project. If a start date is not specified for the project, the schedule is computed as unformatted numerical values with a project start date of 0. The DATE= option can be a SAS date, time, or datetime value (or a number) and can be used to specify a start date for the project. In addition to the start date of the project, you can specify a desired *finish date* for the project using the FBDATE= option.

PROC CPM computes the early start schedule as well as the late start schedule for the project. The project start date is used as the starting point for the calculation of the early start schedule, while the project completion date is used in the computation of the late start schedule. The early start time (E_START) for all *start* activities (those activities with no predecessors) in the project is set to be equal to the value of the DATE parameter (if the FINISHBEFORE option is not specified). The early finish time (E_FINISH) for each start activity is computed as E_START $+$ *dur* , where *dur* is the activity's duration (as specified in the Activity data set). For each of the other activities in the network, the early start time is computed as the maximum of the early finish time of all its immediate predecessors.

The project finish time is computed as the maximum of the early finish time of all the activities in the network. The late finish time (L_FINISH) for all the *finish* activities (those activities with no successors) in the project is set to be equal to the project finish time. The late start time (L_START) is computed as L_FINISH $-$ *dur*. For each of the other activities in the network, the late finish time is computed as the minimum of the late start time of all its immediate successors. If the FIXFINISH option is specified, the late finish time for each finish activity is set to be equal to its early finish time. In other words, the finish activities are not allowed to float to the end of the project.

Once the early and late start schedules have been computed, the procedure computes the free and total float times for each activity. Free float (F_FLOAT) is defined as the maximum delay that can be allowed in an activity without delaying a successor activity. Total float (T_FLOAT) is calculated as the difference between the activity's late finish time and early finish time; it indicates the amount of time by which an activity can be delayed without delaying the entire project. The values of both the float variables are calculated in units of the INTERVAL parameter.

An activity that has zero T_FLOAT is said to be *critical*. As a result of the forward and backward pass computations just described, there is at least one path in the project network that contains only critical activities. This path is called the *critical path*. The duration of the project is equal to the length of the critical path.

If the FBDATE= option is also specified, the project finish time is set equal to the value of the FBDATE= option. The backward pass computation is initiated by setting the late finish time for all the finish activities in the project to be equal to *fbdate*. If the project finish time, as computed from the forward pass calculations, is different from *fbdate*, the longest path in the network may no longer have 0 total float. In such a situation, the critical path is defined to be the path in the network with the least total float. Activities with negative T_FLOAT are referred to as *supercritical* activities.

NOTE: An important requirement for a project network is that it should be *acyclic* (cycles are not allowed).

A network is said to contain a *cycle* (or *loop*) if the precedence relationships starting from an activity loops back to the same activity. The forward and backward pass computations cannot be performed for a cyclic network. If the project network has a cycle, the CPM procedure stops processing after identifying the cycle.

Using the INTERVAL= Option

The INTERVAL= option enables you to define the units of the DURATION variable; that is, you can indicate whether the durations are specified as hours, minutes, days, or in terms of workdays, and so on. In addition to specifying the units, the INTERVAL= option also indicates whether the schedule is to be output as SAS time, date, or datetime values, or as unformatted numeric values.

The prefix *DT* in the value of the INTERVAL= option (as in DTDAY, DTWEEK, and so on) indicates to PROC CPM that the schedule is output as SAS datetime values, and the DATE= option is expected to be a SAS datetime value. Thus, use DTYEAR, DTMONTH, DTQTR, or DTWEEK instead of the corresponding YEAR, MONTH, QTR, or WEEK if the DATE= option is specified as a SAS datetime value.

The start and finish times for the different schedules computed by PROC CPM denote the first and last *day* of work, respectively, when the values are formatted as SAS *date* values. If the times are SAS *time* or *datetime* values, they denote the first and last *second* of work, respectively.

If the INTERVAL= option is specified as WORKDAY, the procedure schedules work on weekdays and nonholidays starting at 9 a.m. and ending at 5 p.m. If you use INTERVAL=DTWRKDAY, the procedure also schedules work only on weekdays and nonholidays. In this case, however, the procedure expects the DATE= option to be a SAS datetime value, and the procedure interprets the start of the workday from the time portion of that option. To change the length of the workday, use the DAYLENGTH= option in conjunction with INTERVAL=DTWRKDAY.

The procedure sets the default value of the INTERVAL= option on the basis of the units of the DATE= option. Table 4.4 lists various valid combinations of the INTERVAL= option and the type of the DATE= option (number, SAS time, date or datetime value) and the resulting interpretation of the duration units and the format type of the schedule variables (numbers, SAS time, date or datetime format) output to the Schedule data set. For each DATE type value, the first INTERVAL value is the default. Thus, if the DATE= option is a SAS date value, the default value of the INTERVAL= option is DAY, and so on.

For the first five specifications of the INTERVAL= option in the last part of Table 4.4 (DTDAY , . . . , DTHOUR), the day starts at *daystart* and is *daylength* hours long.

The procedure may change the INTERVAL= specification and the units of the schedule variables to be compatible with the format specification of the ALIGNDATE variable, or the A_START or A_FINISH variables in the Activity data set, or the PERIOD variable in the Resource data set. For example, if *interval* is specified as DAY, but the ALIGNDATE variable contains SAS datetime values, the schedule is computed in SAS datetime values. Similarly, if *interval* is specified as DAY or WEEKDAY, but some of the durations are fractional, the schedule is computed as SAS datetime values.

Table 4.4 INTERVAL= and DATE= Parameters and Units of Duration

DATE Type	INTERVAL	Units of Duration	Format of Schedule Variables
Number		Period	Unformatted
SAS time	HOUR	Hour	SAS time
	MINUTE	Minute	SAS time
	SECOND	Second	SAS time
SAS date	DAY	Day	SAS date
	WEEKDAY	Day (5-day week)	SAS date
	WORKDAY	Day (5-day week: 9-5 day)	SAS datetime
	WEEK	Week	SAS date
	MONTH	Month	SAS date
	QTR	Quarter	SAS date
	YEAR	Year	SAS date
SAS datetime	DTDAY	Day (7-day week)	SAS datetime
	DTWRKDAY	Day (5-day week)	SAS datetime
	DTSECOND	Second	SAS datetime
	DTMINUTE	Minute	SAS datetime
	DTHOUR	Hour	SAS datetime
	DTWEEK	Week	SAS datetime
	DTMONTH	Month	SAS datetime
	DTQTR	Quarter	SAS datetime
	DTYEAR	Year	SAS datetime

Nonstandard Precedence Relationships

A *standard* precedence constraint between two activities (for example, activity A and an immediate successor B) implies that the second activity is ready to start as soon as the first activity has finished. Such a relationship is called a *finish-to-start* relationship with zero lag. Often, you want to specify other types of relationships between activities. For example:

- Activity B can start five days after activity A has started: start-to-start lag of five days.
- Activity B can start three days after activity A has finished: finish-to-start lag of three days.

The AON representation of the network enables you to specify such relationships between activities: use the LAG= option in the SUCCESSOR statement. This enables you to use variables in the Activity data set that specify the type of relationship between two activities and the time lag between the two events involved; you can also specify the calendar to be used in measuring the lag duration. See the section "SUCCESSOR Statement" on page 90 for information about the specification. Example 4.11, "Nonstandard Relationships," in the section "Examples" illustrates a nonstandard precedence relationship.

This section briefly discusses how the computation of the early and late start schedules, described in the section "Scheduling Subject to Precedence Constraints" on page 93, changes in the presence of nonstandard relationships.

For each (predecessor, successor) pair of activities, the procedure saves the lag type, lag duration, and lag calendar information. Suppose that the predecessor is A, the immediate successor is B, the durations of the two activities are *dur*A and *dur*B, respectively, and the activity's early start and finish times are *pes* and *pef*, respectively. Suppose further that the lag type is *lt*, lag duration is *ld*, and lag calendar is *lc*. Recall that the basic forward and backward passes described in the section "Scheduling Subject to Precedence Constraints" on page 93 assume that all the precedence constraints are standard of the type finish-to-start with zero lag. Thus, in terms of the notation just defined, the early start time of an activity is computed as the maximum of *pef* for all the preceding activities. However, in the presence of nonstandard relationships, the predecessor's value used to compute an activity's early start time depends on the lag type and lag value. Table 4.5 lists the predecessor's value that is used to determine the successor's early start time.

Table 4.5 Effect of Lag Duration and Calendar on Early Start Schedule

Lag Type	Definition	Value Used to Compute Successor's E_START
FS	Finish-to-start	$pef + ld$
SS	Start-to-start	$pes + ld$
SF	Start-to-finish	$pes + ld - dur\mathrm{B}$
FF	Finish-to-finish	$pef + ld - dur\mathrm{B}$

The addition of the lag durations (*ld*) is in units following the lag calendar *lc*; the subtraction of *dur*B is in units of the activity B's calendar. The backward pass to determine the late start schedule is modified in a similar way to include lag durations and calendars.

Time-Constrained Scheduling

You can use the DATE= and FBDATE= options in the PROC CPM statement (or the DATE= option in conjunction with the FINISHBEFORE option) to impose start and finish dates on the project as a whole. Often, you want to impose start or finish constraints on individual activities within the project. The ALIGNDATE and ALIGNTYPE statements enable you to do so. For each activity in the project, you can specify a particular date (as the value of the ALIGNDATE variable) and whether you want the activity to start on or finish before that date (by specifying one of several *alignment types* as the value of the ALIGNTYPE variable). PROC CPM uses all these dates in the computation of the early and late start schedules.

The following explanation best illustrates the restrictions imposed on the start or finish times of an activity by the different types of alignment allowed. Let *d* denote the value of the ALIGNDATE variable for a particular activity and let *dur* be the activity's duration. If *minsdate* and *maxfdate* are used to denote the earliest allowed start date and the latest allowed finish date, respectively, for the activity, then Table 4.6 illustrates the values of *minsdate* and *maxfdate* as a function of the value of the ALIGNTYPE variable.

Once the *minsdate* and *maxfdate* dates have been calculated for all of the activities in the project, the values of *minsdate* are used in the computation of the *early start* schedule and the values of *maxfdate* are used in the computation of the *late start* schedule.

Table 4.6 Determining Alignment Date Values with the ALIGNTYPE Statement

Keywords	**Alignment Type**	*minsdate*	*maxfdate*
SEQ	Start equal	d	$d + dur$
SGE	Start greater than or equal	d	$+ infinity$
SLE	Start less than or equal	$- infinity$	$d + dur$
FEQ	Finish equal	$d - dur$	d
FGE	Finish greater than or equal	$d - dur$	$+ infinity$
FLE	Finish less than or equal	$- infinity$	d
MS	Mandatory start	d	$d + dur$
MF	Mandatory finish	$d - dur$	d

For the first six alignment types in Table 4.6, the value of *minsdate* specifies a lower bound on the early start time and the value of *maxfdate* specifies an upper bound on the late finish time of the activity. The early start time (E_START) of an activity is computed as the maximum of its *minsdate* and the early finish times (E_FINISH) of all its predecessors (E_FINISH=E_START + *dur*). If nonstandard relationships are present in the project, the predecessor's value that is used depends on the type of the lag and the lag duration; Table 4.5 in the previous section lists the values used as a function of the lag type. If a target completion date is not specified (using the FBDATE or FINISHBEFORE options), the project completion time is determined as the maximum value of E_FINISH over all of the activities in the project. The late finish time (L_FINISH) for each of the finish activities (those with no successors) is computed as the minimum of its *maxfdate* and the project completion date; late start time (L_START) is computed as L_FINISH − *dur*. The late finish time (L_FINISH) for each of the other activities in the network is computed as the minimum of its *maxfdate* and the times of all its successors.

It is important to remember that the precedence constraints of the network are always respected (for these first six alignment types). Thus, it is possible that an activity that has an alignment constraint of the type SEQ, constraining it to start on a particular date, say d, may not start on the specified date d due to its predecessors not being finished before d. During resource-constrained scheduling, a further slippage in the start date could occur due to insufficient resources. In other words, *the precedence constraints and resource constraints have priority over the time constraints* (as imposed by the ALIGNDATE and ALIGNTYPE statements) in the determination of the schedule of the activities in the network.

The last two alignment types, MS and MF, however, specify *mandatory dates* for the start and finish times of the activities for both the early and late start schedules. These alignment types can be used to schedule activities to start or finish on a given date disregarding precedence and resource constraints. Thus, an activity with the ALIGNTYPE variable's value equal to MS and the ALIGNDATE variable's value equal to d is scheduled to start on d (for the early, late, and resource-constrained schedules) irrespective of whether or not its predecessors are finished or whether or not there are enough resources.

It is possible for the L_START time of an activity to be less than its E_START time if there are constraints on the start times of certain activities in the network (or constraints on the finish times of some successor activities) that make the target completion date infeasible. In such cases, some of the activities in the network have negative values for T_FLOAT, indicating that these activities are supercritical. See Example 4.12, "Activity Time Constraints," for a demonstration of this situation.

Finish Milestones

By default, the start and finish times for the different schedules computed by PROC CPM denote the first and last *day* of work, respectively, when the values are formatted as SAS *date* values. All start times are assumed to denote the beginning of the day and all finish times are assumed to correspond to the end of the day. If the times are SAS *time* or *datetime* values, they denote the first and last *second* of work, respectively. However, for zero duration activities, *both* the start and the finish times correspond to the beginning of the date (or second) specified.

Thus, according to the preceding definitions, the CPM procedure assumes that all milestones are scheduled at the *beginning* of the day indicated by their start times. In other words, the milestones can be regarded as *start* milestones since they correspond to the *beginning* of the time period indicated by their scheduled times.

However, in some situations, you may want to treat the milestones as *finish* milestones.

Consider the following example:

Activity 'A' has a 2-day duration and is followed by a milestone (zero duration) activity, 'B'. Suppose that activity 'A' starts on March 15, 2004. The default calculations by the CPM procedure will produce the following schedule for the two activities:

```
OBS   Activity   Duration  E_START      E_FINISH

 1       A           2     15MAR2004    16MAR2004
 2       B           0     17Mar2004    17MAR2004
```

The start and finish times of the milestone activity, 'B', are interpreted as the beginning of March 17, 2004. In some situations, you may want the milestones to start and finish on the same day as their predecessors. For instance, in this example, you may want the start and finish time of activity 'B' to be set to March 16, 2004, with the interpretation that the time corresponds to the *end* of the day. Such milestones will be referred to as *finish milestones*.

The SETFINISHMILESTONE option in the PROC CPM statement indicates that a milestone that is linked to its predecessor by a *Finish-to-Start* or a *Finish-to-Finish* precedence constraint should be treated as a *finish milestone*. In other words, such a milestone should have the start and finish time set to the *end* of the day that the predecessor activity finishes. There are some exceptions to this rule:

- There is an alignment constraint on activity 'B' that requires the milestone to start on a later day than the date dictated by the precedence constraint.
- Activity 'B' has an actual start or finish time specified that is inconsistent with the predecessor's finish date.

The alignment constraint that affects the early schedule of the project may not have any impact on the late schedule. Thus, a milestone may be treated as a finish milestone for the late schedule even if it is not a finish milestone according to the early schedule. See Example 4.28 for an illustration of this situation. In addition, while computing the resource-constrained schedule, a start milestone (according to the early schedule) may in fact turn out to be a finish milestone according to the resource-constrained schedule.

Since the same milestone could be treated as either a start or a finish milestone depending on the presence or absence of an alignment constraint, or depending on the type of the schedule (early, late, resource-constrained, or actual), the CPM procedure adds extra variables to the Schedule data set corresponding to each type of schedule. These variables, EFINMILE, LFINMILE, SFINMILE, and AFINMILE, indicate for each milestone activity in the project whether the corresponding schedule times (early, late, resource-constrained, or actual) are to be interpreted as finish milestone times. These variables have a value of '1' if the milestone is treated as a finish milestone for the corresponding schedule; otherwise, the value is missing. In addition to providing an unambiguous interpretation for the schedule times of the milestones, these variables are useful in plotting the schedules correctly using the Gantt procedure. (See Example 4.28).

OUT= Schedule Data Set

The Schedule data set always contains the variables in the Activity data set that are listed in the TAILNODE, HEADNODE, ACTIVITY, SUCCESSOR, DURATION, and ID statements. If the INTPER= option is specified in the PROC CPM statement, then the values of the DURATION variable in the Schedule data set are obtained by multiplying the corresponding values in the Activity data set by *intper*. Thus, the values in the Schedule data set are the durations used by PROC CPM to compute the schedule. If the procedure is used without specifying a RESOURCEIN= data set and only the unconstrained schedule is obtained, then the Schedule data set contains six new variables named E_START, L_START, E_FINISH, L_FINISH, T_FLOAT, and F_FLOAT.

If a resource-constrained schedule is obtained, however, the Schedule data set contains two new variables named S_START and S_FINISH; the T_FLOAT and F_FLOAT variables are omitted. You can request the omission of the E_START and E_FINISH variables by specifying NOE_START and the omission of the L_START and L_FINISH variables by specifying NOL_START in the RESOURCE statement. The variables listed in the RESOURCE statement are also included in the Schedule data set; to omit them, use the NORESOURCEVARS option in the RESOURCE statement. If the DELAYANALYSIS option is specified, the Schedule data set also includes the variables R_DELAY, DELAY_R and SUPPL_R.

If resource-driven durations or resource calendars are in effect, the start and finish times shown in the Schedule data set are computed as the minimum of the start times for all resources for that activity and the maximum of the finish times for all resources for that activity, respectively. For details see the section "Resource-Driven Durations and Resource Calendars" on page 112.

If an ACTUAL statement is specified, the Schedule data set also contains the four variables A_START, A_FINISH, A_DUR, and STATUS.

The format of the schedule variables in this data set (namely, A_START, A_FINISH, E_START, E_FINISH, L_START, and so on) is consistent with the format of the DATE= specification and the INTERVAL= option in the PROC CPM statement.

Definitions of Variables in the OUT= Data Set

Each observation in the Schedule data set is associated with an activity. The variables in the data set have the following meanings.

A_DUR

specifies the actual duration of the activity. This variable is included in the Schedule data set only if the ACTUAL statement is used. The value for this variable is missing unless the activity is completed and may be different from the duration of the activity as specified by the DURATION variable. It is based on the values of the progress variables. See the section "Progress Updating" on page 109 for further details.

A_FINISH

specifies the actual finish time of the activity, either as specified in the Activity data set or as computed by PROC CPM on the basis of the progress variables specified. This variable is included in the Schedule data set only if the ACTUAL statement is used.

A_START

specifies the actual start time of the activity, either as specified in the Activity data set or as computed by PROC CPM on the basis of the progress variables specified. This variable is included in the Schedule data set only if the ACTUAL statement is used.

E_FINISH

specifies the completion time if the activity is started at the early start time.

E_START

specifies the earliest time the activity can be started. This is the maximum of the *maximum* early finish time of all predecessor activities and any lower bound placed on the start time of this activity by the alignment constraints.

F_FLOAT

specifies the free float time, which is the difference between the early finish time of the activity and the minimum early start time of the activity's immediate successors. Consequently, it is the maximum delay that can be tolerated in the activity without affecting the scheduling of a successor activity. The values of this variable are calculated in units of the INTERVAL= parameter.

L_FINISH

specifies the latest completion time of the activity. This is the minimum of the *minimum* late start time of all successor activities and any upper bound placed on the finish time of the activity by the alignment constraints.

L_START

specifies the latest time the activity can be started. This is computed from the activity's latest finish time.

S_FINISH

specifies the resource-constrained finish time of the activity. If resources are insufficient and the procedure cannot schedule the activity, the value is set to missing, unless the FILLUNSCHED option is specified.

S_START

specifies the resource-constrained start time of the activity. If resources are insufficient and the procedure cannot schedule the activity, the value is set to missing, unless the FILLUNSCHED option is specified.

STATUS

specifies the current status of the activity. This is a character valued variable. Possible values for the status of an activity are *Completed*, *In Progress*, *Infeasible* or *Pending*; the meanings are self-evident. If the project is scheduled subject to resource constraints, activities that are *Pending* are classified as *Pending* or *Infeasible* depending on whether or not PROC CPM is able to determine a resource-constrained schedule for the activity.

T_FLOAT

specifies the total float time, which is the difference between the activity late finish time and early finish time. Consequently, it is the maximum delay that can be tolerated in performing the activity and still complete the project on schedule. An activity is said to be on the critical path if T_FLOAT=0. The values of this variable are calculated in units of the INTERVAL= parameter.

If activity splitting is allowed during resource-constrained scheduling, the Schedule data set may contain more than one observation corresponding to each observation in the Activity data set. It will also contain the variable SEGMT_NO, which is explained in the section "Activity Splitting" on page 122.

If the PROJECT statement is used, some additional variables are added to the output data set. See the section "Schedule Data Set" on page 134 for details.

Multiple Calendars

Work pertaining to a given activity is assumed to be done according to a particular *calendar*. A calendar is defined here in terms of a work pattern for each day and a work week structure for each week. In addition, each calendar may have holidays during a given year.

You can associate calendars with Activities (using the CALID variable in the Activity data set) or Resources (using observations with the keyword 'CALENDAR' for the OBSTYPE variable in the Resource data set).

PROC CPM enables you to define very general calendars using the WORKDATA, CALEDATA, and HOLIDATA data sets and options in the PROC CPM statement. Recall that these data sets are referred to as the Workday, Calendar, and Holiday data sets, respectively. The Workday data set specifies distinct shift patterns during a day. The Calendar data set specifies a typical work week for any given calendar; for each day of a typical week, it specifies the shift pattern that is followed. The Holiday data set specifies a list of holidays and the calendars that they refer to; holidays are defined either by specifying the start of the holiday and its duration in *interval* units, or by specifying the start and end of the holiday period. The Activity data set (the DATA= input data set) then specifies the calendar that is used by each activity in the project through the CALID variable (or a default variable _CAL_). Each of the three data sets used to define calendars is described in greater detail later in this section.

Each new value for the CALID variable in either the Calendar data set or the Holiday data set defines a new calendar. If a calendar value appears in the Calendar data set and not in the Holiday data set, it is assumed to have the same holidays as the default calendar (the default calendar is defined later in this section). If a calendar value appears in the Holiday data set and not in the Calendar data set, it is assumed to have the same work pattern structures (for each week and within each day) as the default calendar. In the Activity data set, valid values for the CALID variable are those that are defined in either the Calendar data set or the Holiday data set.

Cautions

The Holiday, Calendar, and Workday data sets and the processing of holidays and different calendars are supported only when *interval* is DAY, WEEKDAY, DTDAY, WORKDAY, DTWRKDAY, DTHOUR, DTMINUTE, or DTSECOND. PROC CPM uses default specifications whenever some information required to define a calendar is missing or invalid. The defaults have been chosen to provide consistency among different types of specifications and to correct for errors in input, while maintaining compatibility with earlier versions of PROC CPM. You get a wide range of control over the calendar specifications, from letting PROC CPM define a single calendar entirely from defaults, to defining several calendars of your choice with precisely defined work patterns for each day of the week and for each week. If the Calendar, Workday, and Holiday data sets are used along with multiple calendar specifications, it is important to remember how all of the data sets and the various options interact to form the work patterns for the different calendars.

Default Calendar

The default calendar is a special calendar that is defined by PROC CPM; its definition and uses are explained in this subsection.

If there is no CALID variable and no Calendar and Workday data sets, the default calendar is defined by *interval* and the DAYSTART= and DAYLENGTH= options in the PROC CPM statement. If *interval* is DAY, DTDAY, DTHOUR, DTMINUTE or DTSECOND, work is done on all seven days of the week; otherwise, Saturday and Sunday are considered to be non-working days. Further, if the schedule is computed as SAS datetime values, the length of the working day is determined by *daystart* and *daylength*. All of the holidays specified in the Holiday data set refer to this default calendar, and all of the activities in the project follow it. Thus, if there is no CALID variable, the default calendar is the only calendar that is used for all of the activities in the project.

If there is a CALID variable that identifies distinct calendars, you can use an observation in the Calendar data set to define the work week structure for the default calendar. Use the value '0' (if CALID is a numeric variable) or the value 'DEFAULT' (if CALID is a character variable) to identify the default calendar. In the absence of such an observation, the default calendar is defined by *interval*, *daystart*, and *daylength*, as described earlier. The default calendar is used to substitute default work patterns for missing values in the Calendar data set or to set default work week structures for newly defined calendars in the Holiday data set.

WORKDATA Data Set

All numeric variables in the Workday data set are assumed to denote unique shift patterns during one working day. For each variable the observations specify, alternately, the times when consecutive shifts start and end. Suppose S1, S2, and S3 are numeric variables formatted as TIME6. Consider the following Workday data:

```
   S1        S2        S3

 7:00         .      7:00    (start)
11:00     08:00     11:00    (end)
12:00         .         .    (start)
16:00         .         .    (end)
```

The variables S1, S2, and S3 define three different work patterns. A missing value in the first observation is assumed to be 0 (or 12:00 midnight); a missing value in any other observation is assumed to denote 24:00 *and ends the definition of the shift*. Thus, the workdays defined are:

- S1 defines a workday starting at 7:00 a.m. and continuing until 4:00 p.m. with an hour off for lunch from 11:00 a.m. until 12:00 noon.

- S2 defines a workday from midnight to 8:00 a.m.
- S3 defines a workday from 7:00 a.m. to 11:00 a.m.

The last two values for the variables S2 and S3 (both values are '24:00', by default) are ignored. This data set can be used to define all of the unique shift patterns that occur in any of the calendars in the project. These shift patterns are tied to the different calendars in which they occur using the Calendar data set.

CALEDATA Data Set

The Calendar data set defines specific calendars using the names of the shift variables in the Workday data set. You can use the variable specified in the CALID statement or a variable named _CAL_ to identify the calendar name or number. Character variables named _SUN_, _MON_, _TUE_, _WED_, _THU_, _FRI_, and _SAT_ are used to indicate the work pattern that is followed on each day of the week. Valid values for these variables are 'HOLIDAY', 'WORKDAY' or, any shift variable name defined in the Workday data set.

NOTE: A missing value for any of these variables is assumed to denote that the work pattern for the corresponding day is the same as for the default calendar.

When *interval* is specified as DTDAY, WORKDAY, or DTWRKDAY, it is necessary to know the length of a *standard* working day in order to be able to compute the schedules consistently. For example, a given calendar may have an eight-hour day on Monday, Tuesday, and Wednesday and a seven-hour day on Thursday and Friday. If a given activity following that calendar has a duration of four days, does it mean that its duration is equal to $8 \times 4 = 32$ hours or $7 \times 4 = 28$ hours? To avoid ambiguity, a numeric variable named D_LENGTH can be specified in the Calendar data set to define the length of a standard working day for the specified calendar. If this variable is not found in the Calendar data set, all calendars for the project are assumed to have a standard daylength as defined by the default calendar.

For example, consider the following Calendar data:

```
_CAL_    _SUN_      _MON_    _TUE_    _FRI_  _SAT_    D_LENGTH

  1      HOLIDAY     S1       S1       S2     S3        8:00
  2      HOLIDAY      .        .        .  HOLIDAY       .
  3         .         .        .        .     .          .
```

These three observations define three calendars: '1', '2', and '3'. The values 'S1', 'S2', and 'S3' refer to the shift variables defined in the section "WORKDATA Data Set" on page 102. Activities in the project can follow either of these three calendars or the default calendar.

Suppose *daystart* has been specified as 9:00 a.m. and *daylength* is eight hours. Further, suppose that *interval* is DTDAY. Using these parameter specifications, PROC CPM defines the default calendar and calendars 1, 2 and 3 using the Calendar data set just defined:

- The default calendar (not specified explicitly in the Calendar data set) is defined using *interval*, *daystart*, and *daylength*. It follows a seven-day week with each day being an eight-hour day (from 9:00 a.m. to 5:00 p.m.). Recall that the default calendar is defined to have seven or five working days depending on whether *interval* is DTDAY or WORKDAY, respectively.

- Calendar ‘1’ (defined in observation 1) has a holiday on Sunday; on Monday and Tuesday work is done from 7:00 a.m. to 11:00 a.m. and then from 12:00 noon to 4:00 p.m.; work on Friday is done from 12:00 (midnight) to 8:00 a.m.; work on Saturday is done from 7:00 a.m. to 11:00 a.m.; on other days work is done from 9:00 a.m. to 5:00 p.m., as defined by the default calendar. The value of D_LENGTH specifies the number of hours in a standard work day; when durations of activities are specified in terms of number of workdays, then the value of D_LENGTH is used as a multiplier to convert workdays to the appropriate number of hours.

- Calendar ‘2’ (defined in observation 2) has holidays on Saturday and Sunday, and on the remaining days, it follows the standard working day as defined by the default calendar.

- Calendar ‘3’ (defined in observation 3) follows the same definition as the default calendar.

NOTE: If there are multiple observations in the Calendar data set identifying the same calendar, all except the first occurrence are ignored. The value ‘0’ (if CALID is a numeric variable) or the value ‘DEFAULT’ (if CALID is a character variable) refers to the default calendar. A missing value for the CALID variable is also assumed to refer to the default calendar. The Calendar data set can be used to define the default calendar also.

HOLIDATA Data Set

The HOLIDATA data set (referred to as the Holiday data set) defines holidays for the different calendars that may be used in the project. Holidays are specified by using the HOLIDAY statement. See the HOLIDAY statement earlier in this chapter for a description of the syntax. This data set must contain a variable (the HOLIDAY variable) whose values specify the start of each holiday. Optionally, the data set may also contain a variable (the HOLIDUR variable) used to specify the length of each holiday or another variable (the HOLIFIN variable) specifying the finish time of each holiday. The variable specified by the CALID statement (or a variable named _CAL_) can be used in this data set to identify the calendar to which each holiday refers. A missing value for the HOLIDAY variable in an observation causes that observation to be ignored. If both the HOLIDUR and the HOLIFIN variables have missing values in a given observation, the holiday is assumed to start at the date and time specified for the HOLIDAY variable and last one unit of *interval* where the INTERVAL= option has been specified as *interval*. If a given observation has valid values for both the HOLIDUR and HOLIFIN variables, only the HOLIFIN variable is used so that the holiday is assumed to start and end as specified by the HOLIDAY and HOLIFIN variables, respectively. A missing value for the CALID variable causes the holiday to be included in all of the calendars, including the default.

The HOLIDUR variable is a natural way of expressing vacation times as *n workdays,* and the HOLIFIN variable is more useful for defining standard holiday periods, such as the CHRISTMAS holiday from 24DEC03 to 26DEC03 (both days inclusive). The HOLIDUR variable is assumed to be in units of *interval* and the procedure uses the particular work pattern structure for the given calendar to compute the length (finish time) of the holiday.

For example, consider the following Holiday data:

```
HOLISTA      HOLIDUR      HOLIFIN      _CAL_

24DEC03         .         26DEC03        .
01JAN04         1            .           1
19JAN04         .            .           2
29JAN04         3            .           2
29JAN04         3            .           3
```

Suppose calendars ‘1’, ‘2’, and ‘3’ and the default calendar have been defined as described earlier in the description of the Calendar and Workday data sets. Recall that in this example INTERVAL=DTDAY, DAYSTART=‘09:00’T, and DAYLENGTH=‘08:00’T. Because the schedule is computed as SAS datetime values (since INTERVAL=DTDAY), the holiday values (specified here as SAS date values) are converted to SAS datetime values. The first observation in the Holiday data set has a missing value for _CAL_ and, hence, the holiday in this observation pertains to all the calendars. As defined by the Holiday data, the holiday lists for the different calendars (not including breaks due to shift definitions) are as shown in Table 4.7.

Even though both calendars ‘2’ and ‘3’ have the same specifications for HOLISTA and HOLIDUR, the actual holiday periods are different for the two calendars. For calendar ‘2’, the three days starting from Thursday, January 29, imply that the holidays are on Thursday, Friday, and Monday (because Saturday and Sunday are already holidays). For calendar ‘3’ (all seven days are working days), the holidays are on Thursday, Friday, and Saturday.

Table 4.7 Holiday Definitions

Calendar	Holiday Start	Holiday End
0	24DEC03:09:00	26DEC03:16:59:59
1	24DEC03:09:00	26DEC03:07:59:59
	01JAN04:00:00	01JAN04:07:59:59
2	24DEC03:09:00	26DEC03:16:59:59
	19JAN04:09:00	19JAN04:16:59:59
	29JAN04:09:00	02FEB04:16:59:59
3	24DEC03:09:00	26DEC03:16:59:59
	29JAN04:09:00	31JAN04:16:59:59

You can use the GANTT procedure to visualize the breaks and holidays for the different calendar. Figure 4.4 shows all the breaks and holidays for the period between Christmas and New Year. Holidays and breaks are denoted by *. Likewise, Figure 4.5 shows the vacation periods in January for calendars ‘2’ and ‘3’.

Figure 4.4 Christmas and New Year Holidays for Multiple Calendars

```
                      Christmas and New Year Holidays

         DEC         DEC         DEC         DEC         DEC         DEC

         22          22          23          23          24          24

   _cal_ 00:00       12:00       00:00       12:00       00:00       12:00

        -+-----------+-----------+-----------+-----------+-----------+-

    0   |*********--------****************--------********************|

    1   |*******----*----***************----*----*********************|

    2   |*********--------****************--------********************|

    3   |*********--------****************--------********************|

        -+-----------+-----------+-----------+-----------+-----------+-


                      Christmas and New Year Holidays

 DEC         DEC         DEC         DEC         DEC         DEC         DEC

 24          25          25          26          26          27          27

 12:00       00:00       12:00       00:00       12:00       00:00       12:00

-+-----------+-----------+-----------+-----------+-----------+-----------+-

|*********************************************************************----|

|*******************************************************************----**|

|*************************************************************************|

|*********************************************************************----|

-+-----------+-----------+-----------+-----------+-----------+-----------+-
```

Figure 4.4 *continued*

```
                         Christmas and New Year Holidays

 DEC         DEC         DEC         DEC         DEC         DEC         DEC
 27          28          28          29          29          30          30
 12:00       00:00       12:00       00:00       12:00       00:00       12:00
-+-----------+-----------+-----------+-----------+-----------+-----------+-
|-----****************--------****************--------****************----|
|*******************************************----*----***************----*-|
|*********************************************--------****************----|
|-----****************--------****************--------****************----|
-+-----------+-----------+-----------+-----------+-----------+-----------+-
```

```
                         Christmas and New Year Holidays

 DEC         DEC         DEC         JAN         JAN         JAN         JAN
 30          31          31          01          01          02          02
 12:00       00:00       12:00       00:00       12:00       00:00       12:00
-+-----------+-----------+-----------+-----------+-----------+-----------+-
|-----****************--------****************--------****************----|
|----*****************--------*******************************--------*****|
|-----****************--------****************--------****************----|
|-----****************--------****************--------****************----|
-+-----------+-----------+-----------+-----------+-----------+-----------+-
```

Figure 4.5 Vacation Time for Calendars 2 and 3

```
                    Vacation Times for Calendars 2 and 3

        JAN        JAN        JAN        JAN        JAN        JAN        JAN
        18         18         19         19         20         20         21
_cal_   00:00      12:00      00:00      12:00      00:00      12:00      00:00
       -+-----------+-----------+-----------+-----------+-----------+-----------+-
  2    |*********************************************************--------********|
  3    |*********--------****************--------****************--------********|
       -+-----------+-----------+-----------+-----------+-----------+-----------+-

                    Vacation Times for Calendars 2 and 3

JAN        JAN        JAN        JAN        JAN        JAN        JAN        JAN
21         21         22         22         23         23         24         24
00:00      12:00      00:00      12:00      00:00      12:00      00:00      12:00
-+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-
|*********--------****************--------****************--------********************|
|*********--------****************--------****************--------****************----|
-+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-

                    Vacation Times for Calendars 2 and 3

JAN        JAN        JAN        JAN        JAN        JAN        JAN        JAN
24         25         25         26         26         27         27         28
12:00      00:00      12:00      00:00      12:00      00:00      12:00      00:00
-+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-
|*********************************************--------****************--------********|
|-----****************--------****************--------****************--------********|
-+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-
```

Figure 4.5 *continued*

Vacation Times for Calendars 2 and 3

```
 JAN         JAN         JAN         JAN         JAN         JAN         JAN         JAN
 28          28          29          29          30          30          31          31
 00:00       12:00       00:00       12:00       00:00       12:00       00:00       12:00
-+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-
|*********--------********************************************************************|
|*********--------********************************************************************|
-+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-
```

Vacation Times for Calendars 2 and 3

```
 JAN         FEB         FEB         FEB         FEB         FEB         FEB         FEB
 31          01          01          02          02          03          03          04
 12:00       00:00       12:00       00:00       12:00       00:00       12:00       00:00
-+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-
|*********************************************************************--------********|
|*********************--------****************--------****************--------********|
-+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-
```

Baseline and Target Schedules

An important aspect of project management is to examine the effects of changing some of the parameters of the project on project completion time. For example, you may want to examine different scenarios by changing the durations of some of the activities, or increasing or decreasing the resource levels. To see the effect of these changes, you need to compare the schedules corresponding to the changes. The BASELINE statement enables you to save a particular schedule as a target schedule and then compare a new schedule against that. See the section "BASELINE Statement" on page 74 for a description of the syntax.

Progress Updating

Once a project has been defined with all of its activities and their relationships, the durations, the resources needed, and so on, it is often useful to monitor its progress periodically. During resource-constrained scheduling, it is useful to schedule only activities that have not yet started, taking into consideration the activities that have already been completed or scheduled and the resources that have already been used by them or allotted for them. The ACTUAL statement is used in PROC CPM to convey information about the current status of a project. As information about the activities becomes available, it can be incorporated into the schedule of the project through the specification of the actual start or finish times or both, the duration that is still remaining for the activity, or the percentage of work that has been completed on an activity. The specification of the progress variables and the options in the ACTUAL statement have been described earlier in this chapter. This section describes how the options work together and how some default values are determined.

The following options are discussed in this section:

- the TIMENOW= option
- the AUTOUPDT and NOAUTOUPDT options
- the TIMENOWSPLT option
- the progress variables (A_START, A_FINISH, REMDUR, and PCTCOMP)

The TIMENOW= option is specified in the ACTUAL statement. The value of the TIMENOW= option (often referred to simply as TIMENOW) is used as a reference point to resolve the values of the remaining duration and percent completion times. All actual start and finish times specified are checked to ensure that they are less than TIMENOW. If there is some inconsistency, a warning message is printed to the log.

If the ACTUAL statement is used, at least one of the four progress variables must be specified. PROC CPM uses the nonmissing values for the progress variables in any given observation to determine the information that is to be used for the activity. It is possible that there are some inconsistencies in the specification of the values relating to the progress information. For example, an activity may have valid values for both the A_START and the A_FINISH variables and also have the value of the PCTCOMP variable less than 100. PROC CPM looks at the values in a specific order, resolving inconsistencies in a reasonable manner. Further, PROC CPM determines revised estimates of the durations of the activities on the basis of the actual information.

Suppose that for a given activity, *as* is the actual start, *af* is the actual finish, *remdur* is the remaining duration, *pctc* is the percent complete, and *dur* is the duration of the activity as specified by the values of the corresponding variables in the Activity data set. (If a particular variable is not specified, assume that the corresponding value is missing.)

The *elapsed duration* of an activity in progress is the time lapse between its actual start and TIMENOW; the *revised duration* of the activity is the *updated duration* of the activity that is used to calculate the projected finish time for activities in progress and the *actual duration* for activities that are completed. The *revised duration* is used by PROC CPM to compute the updated schedule as described later in this section. In the discussion that follows, *as*, *af*, *remdur*, and *pctc* refer to the *actual start time*, *actual finish time*, *remaining duration*, and *percent completed*, respectively, for the activity in the Activity data set, while A_START, A_FINISH, and A_DUR refer to the values calculated by PROC CPM for the corresponding new variables added to the Schedule data set.

The following is a list of some of the conventions used by PROC CPM in calculating the *revised duration*:

- If both *as* and *af* are specified, the *revised duration* is computed as the time, excluding non-working periods, between *as* and *af*; in the Schedule data set, the variable A_DUR is also set to this value; A_START is set to *as* and A_FINISH to *af*.

- If *as* is specified without *af*, PROC CPM uses *remdur* to compute the *revised duration* as the sum of the elapsed duration and the remaining duration.

- If *as* is specified and both *af* and *remdur* are missing, the *revised duration* is computed on the basis of the elapsed duration and *pctc*.

- If *as* is specified and *af*, *remdur* and *pctc* are not specified, the duration is not revised. If the time lapse between *as* and TIMENOW is greater than or equal to the duration of the activity, it is assumed to have finished at the appropriate time ($as + dur$) and the Schedule data set has the appropriate values for A_START, A_FINISH, and A_DUR.

- If *as* is missing and *af* is valid, PROC CPM determines *as* on the basis of *af* and the specified duration (*remdur* and *pctc*, if specified, are ignored.)

- If *as* and *af* are both missing, the *revised duration* is determined on the basis of *remdur* and *pctc*. If the activity has started (if $pctc > 0$ or $remdur < dur$), *as* is set appropriately, and if it has also finished (which is the case if $pctc = 100$ or $remdur = 0$), *af* is also set.

Using the preceding rules, PROC CPM attempts to determine actual start and finish times for as many activities as possible using the information given for each activity. The next question is: What about activities that have missing values for the actual start and finish times? Suppose a given activity has a valid value for A_START and is currently in progress. It seems logical for successors of this activity to have missing values for A_START. But how about predecessors of the activity? If they have missing values for A_START and A_FINISH, does it mean that there was an error in the input of the actual dates or an error in the precedence constraints? The AUTOUPDT and NOAUTOUPDT options enable you to control the answer to this question. AUTOUPDT instructs CPM to automatically fill in appropriate A_START and A_FINISH values for all activities that precede activities which have already started. NOAUTOUPDT implies that only those activities that have explicit progress information confirming their status are assumed to be in progress or completed; all other activities are assumed to have an implicit start date that is greater than or equal to TIMENOW. In other words, NOAUTOUPDT assumes that the precedence constraints may be overridden by the actual data. The default option is AUTOUPDT.

The scheduling algorithm treats the actual start and finish times as follows:

- If A_START is not missing, the E_START time is set equal to A_START during the forward pass, and the E_FINISH time is set equal to E_START + the *revised duration*.

- If A_START is missing, the E_START time is computed as before.

- If A_FINISH or A_START is not missing, the L_FINISH time is set equal to A_FINISH during the backward pass, and the L_START time is computed on the basis of L_FINISH and the *revised duration*.

  This rule causes the late start schedule to be the same as the early start schedule for completed or in-progress activities. Thus, T_FLOAT and F_FLOAT are 0 for such activities. Use the SHOWFLOAT option if you want to allow nonzero float for in-progress or completed activities. In this case, the late start schedule is computed as before, using the precedence constraints, so that you can determine the degree of lateness for the activities that have already been completed or are in progress.

- If E_START is less than TIMENOW for an activity (and thus it is also the same as A_START), the activity is scheduled during resource allocation even if there are not enough resources (a warning message is printed to the log if this is the case). Thus, resource-constrained scheduling is done only for the period starting from TIMENOW.

  NOTE: The resources required by activities that are completed or in progress are accounted for and the corresponding changes are made to the resource availability profile before starting the constrained scheduling process at TIMENOW.

- If resource-constrained scheduling is being performed, the TIMENOWSPLT option can be used. This option affects those activities that are currently in progress that cause resource infeasibilities. The TIMENOWSPLT option causes such activities to be split at TIMENOW into segments; the first segment is assumed to be complete before TIMENOW, and the second segment is delayed until sufficient resources are available.

The Schedule data set contains the actual start times (A_START) for all activities that are in progress or completed and the actual finish times (A_FINISH) and the actual duration times (A_DUR) for all activities that are completed. Some of these values may have been derived from the percent completion or remaining duration times in the Activity data set or may have been implicitly determined through the AUTOUPDT option. Also included in the Schedule data set is a variable named STATUS describing the status of each activity. The possible values are *Completed*, *In Progress*, *Infeasible*, and *Pending*; the interpretations are self-evident.

If the ESTPCTC option is specified, the Schedule data set also contains a variable named PCT_COMP that contains the percent completion time for each activity in the project.

Resource-Driven Durations and Resource Calendars

The DURATION variable enables you to specify a fixed duration for an activity. The CPM procedure then assumes that all the resources for that activity are required throughout the duration of that activity; further, the activity is assumed to follow the work pattern specified by the activity's calendar. Suppose that there are multiple resources required by an activity, each following a different calendar and each requiring varying amounts of work. For example, a programming task may require 50 hours of a programmer's time and 20 hours of a tester's time. Further, the programmer may work full time on the tasks, while the tester, due to other commitments, may work only half time on the same activity. The scheduling could be further complicated if the tester and the programmer followed different calendars. Situations of this type can be modeled using resource-driven durations and resource calendars.

The WORK variable in the Activity data set specifies the *total* amount of work required by one unit of a resource. Unlike the DURATION variable, which represents a fixed duration for an activity for all its resources, the WORK variable *drives* the duration for each resource required by the activity using the resource rate specified. You can specify different amounts of work for different resources by using different observations to specify rates and total work for the different resources. Consider the following data from an Activity data set:

```
ACT      WORK      PGMR   TESTER

 1        50        1        .
 1        20        .       .5
 2        15        1       1
```

PGMR and TESTER are resource variables specifying the rate at which the respective resource is required (used) for the particular activity; WORK specifies the total number of hours (assuming that the INTERVAL parameter has been specified as HOUR) of work required by each resource that has a rate specified in that observation. Thus, Activity '1' requires 50 hours of the resource PGMR and 20 hours of the resource TESTER, while activity '2' requires 15 hours of each of the two resources. Using the rates for the resources specified in the preceding data, the procedure determines the resource durations for activity 1 to be 50 hours for PGMR and 40 hours for TESTER. Likewise, the resource durations for both resources are 15 hours for activity 2.

In the forward and backward pass calculations, the procedure computes the schedules for each resource and sets the activity's start (finish) time to be the minimum (maximum) of the start (finish) times for all the resources.

Some activities may have a fixed duration for some resources and a resource-driven duration for other resources. For such activities, use the DURATION variable to specify the fixed duration and the WORK variable to specify the total amount of work required for the activity. If a particular observation has values specified for both the WORK and DURATION variables, use the resource type information in the Resource data set (described in the section "RESOURCEIN= Input Data Set" on page 114) to determine if the resource *drives* the duration of the activity.

Recall that the CALID variable in the Activity data set specifies the calendar that is used by each activity in the project. In addition, you can also associate calendars with the resources in the project. Resource calendars are specified in the Resource data set. However, the CALID variable must be numeric for you to associate calendars with resources; in other words, the calendars must be identified by numbers and not names.

Resource Usage and Allocation

Often the activities in a project use several resources. If you assume that these resources are available in unlimited quantities, then the only restrictions on the start and finish times of the activities in the project are those imposed by precedence constraints and dates specified for alignment of the activities. In most practical situations, however, there are limitations on the availability of resources; as a result, neither the early start schedule nor the late start schedule (nor any intermediate schedule for that matter) may be feasible. In such cases, the project manager is faced with the task of scheduling the activities in the project subject to constraints on resource availability in addition to the precedence constraints and constraints on the start and finish times of certain activities in the project. This problem is known as *resource allocation*.

You can use PROC CPM to schedule the activities in a project subject to resource constraints. To perform resource allocation, you must specify the resource requirements for each activity in the project and also specify the amount of resources available on each day under consideration. The resource requirements are given in the Activity data set, with the variable names identified to PROC CPM through the RESOURCE statement. The levels of resources available on different dates, as well as other information regarding the resources, such as the type of resource, the priority of the resource, and so forth, are obtained from the RESOURCEIN= data set.

Specifying resource requirements is described in detail in the section "Specifying Resource Requirements" on page 118, and the description of the format of the Resource data set is given in the section "RESOURCEIN= Input Data Set" on page 114, which follows. the section "Scheduling Method" on page 119 describes how you can use the SCHEDRULE= and DELAY= options (and other options) in conjunction with certain special observations in the Resource data set to control the process of resource allocation to suit your needs.

Subsequent sections describe the different scheduling rules, supplementary resources, activity splitting, progress updating, and alternate resources.

RESOURCEIN= Input Data Set

The RESOURCEIN= data set (referred to as the Resource data set) contains all of the necessary information about the resources that are to be used by PROC CPM to schedule the project. Typically, the Resource data set contains the resource variables (numeric), a type identifier variable (character) that identifies the type of information in each observation, a period variable (numeric and usually a SAS time, date, or datetime variable), and a RESID variable that is used to specify *alternate resources* and *auxiliary resources*.

The value of the type identifier variable in each observation tells CPM how to interpret that observation. Valid values for this variable are RESLEVEL, RESTYPE, RESUSAGE, RESPRTY, SUPLEVEL, ALTPRTY, ALTRATE, RESRCDUR, CALENDAR, MULTALT, MINARATE, and AUXRES.

Table 4.8 Type Identifier Variables

Type Identifier Keywords	**Description**	**Values**
'RESLEVEL'	Contains levels available for each resource from the time specified in the period variable.	Missing values are not allowed. For consumable resources, the observation indicates the total availability and for replenishable resources, the new level.
'RESTYPE'	Specifies the nature of the resources, i.e., if they are consumable, replenishable, replenishable aggregate or consumable aggregate resources.	1 = Replenishable 2 = Consumable 3 = Replenishable Aggregate 4 = Consumable Aggregate
'RESUSAGE'	Indicates a profile of usage for a consumable resource	0 = Resource used continuously at specified rate 1 = Resource required at the beginning of activity 2 = Resource used at the end of activity A missing value is treated as 0.
'RESPRTY'	Specifies that PROC CPM should sort the activities in the waiting list in the order of increasing values of the *resource priority* for the most important resource used by each activity.	Low values indicate high priority.
'SUPLEVEL'	Specifies the amount of extra resource available for use through out the duration of the project.	
'RESRCDUR'	Specifies the effect of the resource on the activity's duration.	0 = Resource uses a fixed duration 1 = Driving resource 2 = Spanning resource
'CALENDAR'	Specifies the calendar that is followed by each resource. If no calendar is specified for a given resource, the relevant activity's calendar is used instead.	Requires the calendar variable in the Activity and other data sets to be numeric.

'MULTALT'	Indicates which resources can have multiple alternate resources.	0 = Multiple alternates not allowed 1 = Multiple alternates allowed
'MINARATE'	Indicates the minimum rate of substitution for each resource, whenever multiple alternates are used.	
'ALTRATE'	Specifies the rate of substitution of alternate resources when a resource has more than one substitute. Resource data set must have a RESID variable.	Lower value indicates higher priority. Missing values imply that the particular resource cannot be substituted.
'ALTPRTY'	Specifies the prioritization of the alternate resources when a resource has more than one substitution. Resource data set must have a RESID variable.	Lower values indicate higher priority.
'AUXRES'	Specifies the auxiliary resources that are needed for each primary resource. RESID variable specifies the name of the primary resource.	Value for each auxiliary resource indicates the rate at which it is required whenever theprimary resource is used.

If the value of the type identifier variable in a particular observation is 'RESLEVEL', then that observation contains the levels available for each resource from the time specified in the period variable. Missing values are not allowed for the period variable in an observation containing the levels of the resources. For consumable resources, the observation indicates the *total availability* and *not the increase in the availability*. Likewise, for replenishable resources, the observation indicates the *new level* and *not the change in the level* of the resource.

Each resource can be classified as either consumable or replenishable. A consumable resource is one that is used up by the job (such as bricks or money), while a replenishable resource becomes available again once a job using it is over (such as manpower or machinery). If the value of the type identifier variable is 'RESTYPE', then that observation identifies the nature (consumable or replenishable) of the resource. The observation contains a value 1 for a replenishable resource and a value 2 for a consumable one. A missing value in this observation is treated as 1. In fact, if there is no observation in the Resource data set with the type identifier variable equal to 'RESTYPE', then all resources are assumed to be replenishable.

Sometimes, it may be useful to include resources in the project that are to be used only for aggregation purposes. You can indicate that a given resource is to be used for aggregation, and not for resource allocation, by specifying the values 3 or 4, depending on whether the resource is replenishable or consumable. In other words, use 3 for replenishable aggregate resources and 4 for consumable aggregate resources.

Consumable resources are assumed to be used continuously throughout the duration of the activity at the rate specified in the Activity data set (as described in the section "Specifying Resource Requirements" on page 118). For example, when you specify a rate of 100 per day for bricks, the CPM procedure assumes that the activity consumes bricks at the constant rate of 100 per day. Sometimes, you may wish to allocate all of the resource at the beginning or end of an activity. For example, you may pay an *advance* at the start of a contracted activity while the full *payment* is made when the activity is completed. You can indicate such a profile of usage for a consumable resource using the keyword 'RESUSAGE' for the value of the type identifier variable. Valid values for the resource variables in such an observation are 0, 1, and 2. A value

0 indicates that the resource is used continuously at the specified rate throughout the activity's duration, a value 1 indicates that the resource is required at the beginning of the activity, and a value 2 specifies that the resource is used at the end of the activity. A missing value in this observation is treated as 0.

One of the scheduling rules that can be specified using the SCHEDRULE= option is RESPRTY, which requires ordering the resources according to some priority (details are given in the section "Scheduling Rules" on page 120). If this option is used, there must be an observation in the Resource data set with the type identifier variable taking the value 'RESPRTY'. This observation specifies the ordering of the resources.

If the type identifier variable is given as 'SUPLEVEL', the observation denotes the amount of extra resource that is available for use throughout the duration of the project. This extra resource is used only if the activity cannot be scheduled without delaying it beyond its late start time. See the section "Secondary Levels of Resources" on page 121 for details about the use of supplementary levels of resources in conjunction with the DELAY= and ACTDELAY= options.

If the type identifier variable is specified as 'ALTRATE', 'ALTPRTY', or 'AUXRES', the Resource data set must also have a RESID variable that is used to identify the name of a resource for which the current observation lists the possible alternate resources or the required auxiliary resources. See the section "Specifying Alternate Resources" on page 123 and the section "Auxiliary Resources" on page 127 for details.

If the value of the type identifier variable is 'RESRCDUR', that observation specifies the effect of the resource on an activity's duration. Valid values for the resource variables in such an observation are 0, 1, and 2. A value 0 indicates that the resource uses a fixed duration (specified by the DURATION variable); in other words, the activity's duration is not affected by changing the rate of the resource. A value 1 indicates that the WORK variable for an activity specifies the total amount of work required by the resource that is used to calculate the time required by the resource to complete its work on that activity; such a resource is referred to as a *driving* resource. The value 2 indicates a third type of resource; such a resource (referred to as a *spanning* resource) is required throughout the activity's duration, no matter which resource is working on it. For example, an activity might require 10 percent of a "supervisor," or the use of a particular room, throughout its duration. For such an activity, the duration used for the spanning resource is computed after determining the span of the activity for all the other resources.

If the value of the type identifier variable is 'CALENDAR', that observation specifies the calendar that is followed by each resource. If no calendar is specified for a given resource, the relevant activity's calendar is used instead. This use of the calendar requires that the calendar variable in the Activity and other data sets be numeric.

If the value of the type identifier variable is 'MULTALT', that observation indicates which resources can have multiple alternate resources. The value 1 for a resource variable in the observation indicates that multiple alternates are allowed for that resource, and a value 0 indicates that multiple alternates are not allowed. See the section "Specifying Multiple Alternates" on page 125 for details.

If the value of the type identifier variable is 'MINARATE', that observation indicates the minimum rate of substitution for each resource, whenever multiple alternates are used. The 'MINARATE' values specified in this observation are used only if the MULTIPLEALTERNATES option is specified or if the Resource data set has an observation with the type identifier value of 'MULTALT'.

The period variable must have nonmissing values for observations specifying the levels of the resources (that is, with type identifier equal to 'RESLEVEL'). However, the period variable does not have any meaning when the type identifier variable has any value other than 'RESLEVEL'; if the period variable has nonmissing values in these observations, it is ignored. The Resource data set must be sorted in order of *increasing* values of the period variable.

Multiple observations are allowed for each type of observation. If there is a conflict in the values specified, only the first nonmissing value is honored; for example, if there are two observations of the type 'RESTYPE' and a resource variable has value 1 in the first and 2 in the second of these observations, the resource type is assumed to be 1 (replenishable). On the other hand, if the value is missing in the first observation but set to 2 in the second, the resource type is assumed to be 2 (consumable).

A resource is available at the specified level from the time given in the first observation with a nonmissing value for the resource. Its level changes (to the new value) whenever a new observation is encountered with a nonmissing value, and the date of change is the date specified in this observation.

The following examples illustrate the details about the Resource data set. Consider the following Resource data:

OBS	OBSTYPE	DATE	WORKERS	BRICKS	PAYMENT	ADVANCE
1	RESTYPE	.	1	2	2	2
2	RESUSAGE	.	.	0	2	1
3	RESPRTY	.	10	10	10	10
4	SUPLEVEL	.	1	.	.	.
5	RESLEVEL	1JUL04	.	1000	2000	500
6	RESLEVEL	5JUL04	4	.	.	.
7	RESLEVEL	9JUL04	.	1500	.	.

There are four resources in these data, WORKERS, BRICKS, PAYMENT, and ADVANCE. The variable OBSTYPE is the type identifier, and the variable DATE is the period variable. The first observation (because OBSTYPE has value 'RESTYPE') indicates that WORKERS is a replenishable resource while the other three resources are consumable. The second observation indicates the usage profile for the consumable resources: the resource BRICKS is used continuously throughout the duration of an activity, while the resource PAYMENT is required at the end of the activity and the resource ADVANCE is needed at the start of the activity. The third observation indicates that all the resources have equal priority. In the fourth observation, a value '1' under WORKERS indicates that a supplementary level of 1 worker is available if necessary, while no reserve is available for the resources BRICKS, PAYMENT, and ADVANCE.

The next three observations indicate the resource availability profile. The resource WORKERS is unavailable until July 5, 2004, when the level jumps from 0 to 4 and remains at that level through the end of the project. The resource BRICKS is available from July 1, 2004, at level 1000, while the resource levels for PAYMENT, and ADVANCE are 2000 and 500, respectively. On July 9, an additional 500 bricks are made available to increase the total availability to 1500. Missing values in observations 5 and 6 indicate that there is no change in the availability for the respective resources.

As another example, suppose that you want to treat BRICKS as an aggregate resource (one that is not to be included in resource allocation). Then consider the following data from a Resource data set:

OBSTYPE	BRICKS	PAINTER	SUPERV
RESTYPE	4	1	1
RESRCDUR	0	1	2
CALENDAR	1	0	0

The first observation indicates that the resource BRICKS is consumable and is to be used only for aggregation while the other two resources are replenishable and are to be treated as constrained resources during resource allocation.

The second observation, with the keyword 'RESRCDUR', specifies the effect of the resource on an activity's duration. The value '0' for the resource BRICKS implies that this resource does not affect the duration of an activity. On the other hand, the value '1' identifies the resource PAINTER as a driving resource; this means that by increasing the number of painters, an activity's duration can be decreased. The procedure uses this information about the nature of the resource only if a particular observation in the Activity data set has valid values for both the WORK and DURATION variables. Otherwise, if you specify a value only for the WORK variable, the procedure assumes that the resource specifications in that observation drive the activity's duration. Likewise, if you specify a value only for the DURATION variable, the procedure assumes that the resources specified in that observation require a fixed duration.

In the Resource data set specifications, the second observation also identifies the resource SUPERV to be of the spanning type. In other words, such a resource is required by an activity whenever any of the other resources are working on the same activity. Thus, if you add more painters to an activity, thereby reducing its duration, the supervisor (a spanning resource) will be needed for a shorter time.

The third observation indicates the calendar to be used in calculating the activity's start and finish times for the particular resource. If you do not specify a calendar, the procedure uses the activity's calendar.

Specifying Resource Requirements

To perform resource allocation or to summarize the resource utilization, you must specify the amount of resources required by each activity. In this section, the format for this specification is described. The amount required by each activity for each of the resources listed in the RESOURCE statement is specified in the Activity data set. The requirements for each activity are assumed to be constant throughout the activity's duration. A missing value for a resource variable in the Activity data set indicates that the particular resource is not required for the activity in that observation.

The interpretation of the specification depends on whether or not the resource is replenishable. Suppose that the value for a given resource variable in a particular observation is 'x'. If the resource is *replenishable,* it indicates that x units of the resource are required throughout the duration of the activity specified in that observation. On the other hand, if the resource is *consumable,* it indicates that the specified resource is consumed at the rate of x units per unit *interval,* where *interval* is the value specified in the INTERVAL= option in the PROC CPM statement. For example, consider the following specification:

```
OBS   ACTIVITY    DUR     WORKERS     BRICKS

 1       A         5         .          100
 2       B         4         2           .
```

Here, ACTIVITY denotes the activity under consideration, DUR is the duration in days (assuming that INTERVAL=DAY), and the resource variables are WORKERS and BRICKS. A missing value for WORKERS in observation 1 indicates that activity 'A' does not need the resource WORKERS, while the same is true for the resource BRICKS and activity 'B'. You can assume that the resource WORKERS has been identified as replenishable, and the resource BRICKS has been identified as consumable in a Resource data set. Thus, a value '100' for the consumable resource BRICKS indicates that 100 bricks per day are required for each of the 5 days of the duration of activity 'A', and a value '2' for the replenishable resource WORKERS indicates that 2 workers are required throughout the duration (4 days) of activity 'B'. Recall that consumable resources can be further identified as having a special usage profile, indicating that the requirement is only at the beginning or end of an activity. See the section "Variable Usage Profile for Consumable Resources" on page 130 for details.

Negative Resource Requirements

The CPM procedure enables you to specify negative resource requirements. A negative requirement indicates that a resource is *produced* instead of *consumed*. Typically, this interpretation is valid only for consumable resources. For example, a brick-making machine may produce bricks at the rate of 1000 units per hour which are then available for consumption by other tasks in the project. To indicate that a resource is produced (and not consumed) by an activity, specify the rate of usage for the resource as a negative number. For example, to indicate that a machine produces boxed cards at the rate of 5000 boxes per day, set the value of the resource, NUMBOXES, to -5000.

Scheduling Method

PROC CPM uses the serial-parallel (serial in time and parallel in activities) method of scheduling. In this section, the basic scheduling algorithm is described. (Modifications to the algorithm if an ACTUAL statement is used, if activity splitting is allowed, or if alternate resources are specified, are described later.) The basic algorithm proceeds through the following steps:

1. An initial tentative schedule describing the early and late start and finish times is determined without taking any resource constraints into account. This schedule does, however, reflect any restrictions placed on the start and finish times by the use of the ALIGNDATE and ALIGNTYPE statements. As much as possible, PROC CPM tries to schedule each activity to start at its E_START time (*e_start*, as calculated in this step). Set *time*=min(*e_start*), where the minimum is taken over all the activities in the network.

2. All of the activities whose *e_start* values coincide with *time* are arranged in a waiting list that is sorted according to the rule specified in the SCHEDRULE= option. (See the section "Scheduling Rules" on page 120 for details about the valid values of this option.) The SCHEDRULE2= option can be used to break ties. PROC CPM tries to schedule the activities in the same order as on this list. For each activity the procedure checks to see if the required amount of each resource will be available throughout the activity's duration; if enough resources are available, the activity is scheduled to start at *time*. Otherwise, the resource availability profile is examined to see if there is likely to be an increase in resources in the future. If none is perceived until *l_start* + *delay*, the procedure tries to schedule the activity to start at *time* using supplementary levels of the resources (if there is an observation in the Resource data set specifying supplementary levels of resources); otherwise, it is postponed. (Note that if the AWAITDELAY option is specified, and there are not enough resources at *time*, the activity is not scheduled at *time* using supplementary resources). If *time* is equal to or greater than the value of *l_start* + *delay*, and the activity cannot be scheduled (even using supplementary resources), PROC CPM stops with an error message, giving a partial schedule. You can also specify a cut-off date (using the STOPDATE= option) when resource constrained scheduling is to stop.

   Once an activity that uses a supplementary level of a replenishable resource is over, the supplementary level that was used is returned to the reservoir and is not used again until needed. For consumable resources, if supplementary levels were used on a particular date, PROC CPM attempts to bring the reservoir back to the original level at the earliest possible time. In other words, the next time the primary availability of the resource increases, the reservoir is first used to replenish the supplementary level of the resource. (See Example 4.16, “Using Supplementary Resources”). Adjustment is made to the resource availability profile to account for any activity that is scheduled to start at *time*.

3. All of the activities in the waiting list that were unable to be scheduled in Step 2 are postponed and are tentatively scheduled to start at the time when the next change takes place in the resource availability profile (that is, their *e_start* is set to the next change date in the availability of resources). *time* is advanced to the minimum *e_start* time of all unscheduled activities.

Steps 1, 2, and 3 are repeated until all activities are scheduled or the procedure stops with an error message.

Some important points to keep in mind are:

- Holidays and other non-working times are automatically accounted for in the process of resource allocation. Do not specify zero availabilities for the resources on holidays; PROC CPM accounts for holidays and weekends during resource allocation just as in the unrestricted case.
- It is assumed that the activities cannot be interrupted once they are started, unless one of the splitting options is used. See the section “Activity Splitting” on page 122 for details.

Scheduling Rules

The SCHEDRULE= option specifies the criterion to use for determining the order in which activities are to be considered while scheduling them subject to resource constraints. As described in the section “Scheduling Method” on page 119, at a given time specified by *time*, all activities whose tentative *e_start* coincides with *time* are arranged in a list ordered according to the scheduling rule, *schedrule*. The SCHEDRULE2= option can be used to break ties caused by the SCHEDRULE= option; valid values for *schedrule2* are the same as for *schedrule*. However, if *schedrule* is ACTPRTY, then *schedrule2* cannot be RESPRTY, and vice versa.

The following is a list of the six valid values of *schedrule*, along with a brief description of their respective effects.

ACTPRTY

specifies that PROC CPM should sort the activities in the waiting list in the order of increasing values of the variable specified in the ACTIVITYPRTY= option in the RESOURCE statement. This variable specifies a user-assigned priority to each activity in the project (low value of the variable indicates high priority).

NOTE: If SCHEDRULE is specified as ACTPRTY, the RESOURCE statement must contain the specification of the variable in the Activity data set that assigns priorities to the activities; if the variable name is not specified through the ACTIVITYPRTY= option, then CPM ignores the specification for the SCHEDRULE= option and uses the default scheduling rule, LST, instead.

DELAYLST

specifies that the activities in the waiting list are sorted in the order of increasing L_START + ACTDELAY, where ACTDELAY is the value of the ACTDELAY variable for that activity.

LFT

specifies that the activities in the waiting list are sorted in the order of increasing L_FINISH time.

LST
specifies that the activities in the waiting list are sorted in the order of increasing L_START time. Thus, this option causes activities that are closer to being critical to be scheduled first. This is the default rule.

RESPRTY
specifies that PROC CPM should sort the activities in the waiting list in the order of increasing values of the *resource priority* for the most important resource used by each activity. In order for this scheduling rule to be valid, there must be an observation in the Resource data set identified by the value RESPRTY for the type identifier variable and specifying priorities for the resources. PROC CPM uses these priority values (once again, low values indicate high priority) to order the activities; then, the activities in the waiting list are ordered according to the highest priority resource used by them. In other words, the CPM procedure uses the resource priorities to assign priorities to the activities in the project; these activity priorities are then used to order the activities in the waiting list (in increasing order). If this option is specified, and there is no observation in the Resource data set specifying the resource priorities, PROC CPM ignores the specification for the SCHEDRULE= option and uses the default scheduling rule, LST, instead.

SHORTDUR
specifies that the activities in the waiting list are sorted in the order of increasing durations. Thus, PROC CPM tries to schedule activities with shorter durations first.

Secondary Levels of Resources

There are two factors that you can use to control the process of scheduling subject to resource constraints: *time* and *resources*. In some applications, time is the most important factor, and you may be willing to use extra resources in order to meet project deadlines; in other applications, you may be willing to delay the project completion by an arbitrary amount of time if insufficient resources warrant doing so. The DELAY= and ACTDELAY= options and the availability of supplementary resources enable you to select either method or a combination of the two approaches.

In the first case, where you do not want the project to be delayed, specify the availability of supplementary resources in the Resource data set and set DELAY=0. In the latter case, where extra resources are unavailable and you are willing to delay project completion time, set the DELAY= option to some very large number or leave it unspecified (in which case it is assumed to be $+$ INFINITY). You can achieve a combination of both effects (using supplementary levels and setting a limit on the delay allowed) by specifying an intermediate value for the DELAY= option and including an observation in the Resource data set with supplementary levels.

You can also use the INFEASDIAGNOSTIC option which is equivalent to specifying infinite supplementary levels for all the resources under consideration. In this case, the DELAY= value is assumed to equal the default value of +INFINITY, unless it is specified otherwise. See Example 4.17, “INFEASDIAGNOSTIC Option and Aggregate Resource Type,” for an illustration.

The DELAY= option presupposes that all the activities can be subjected to the same amount of delay. In some situations, you may want to control the amount of delay for each activity on the basis of some criterion, say the amount of float present in the activity. The ACTDELAY= option enables you to specify a variable amount of delay for each activity.

Resource-Driven Durations and Resource Allocation

If resource-driven durations or resource calendars are specified, the procedure computes the start and finish times for each resource separately for each activity. An activity is considered to be completed only when all the resources have completed their work on that activity. Thus, an activity's start (finish) time is computed as the minimum (maximum) of the start (finish) times for all the resources used by that activity.

During resource-constrained scheduling, an activity enters the list of activities waiting for resources when all its precedence constraints have been satisfied. As before, this list is ordered using the scheduling rule specified. At this point, a tentative start and finish time is computed for each of the resources required by the activity using the resource's duration and calendar. An attempt is made to schedule *all* of this activity's resources at these calculated times using the available resources. If the attempt is successful, the activity is scheduled to start at the given time with the appropriate resource schedule times, and the required resources are reduced from the resource availabilities. Otherwise, the procedure attempts to schedule the next activity in the list of activities waiting for resources. When all activities have been considered at the given time, the procedure continues to the next event and continues the allocation process. Note that, at a given point of time, the procedure schedules the activity only if all the required resources are available for that activity to start at that time (or at the nearest time per that resource's calendar), unless you specify the INDEPENDENTALLOC option.

The INDEPENDENTALLOC option enables each resource to be scheduled independently for the activity. Thus, when an activity enters the list of activities waiting for resources, each resource requirement is considered independently, and a particular resource can be scheduled for that activity even if none of the other resources are available. However, the spanning type of resources must always be available throughout the activity's duration. The activity is considered to be finished (and its successors can start) only after all the resources for that activity have been scheduled. This option is valid even if all activities have fixed durations and calendars are not associated with resources.

Activity Splitting

As mentioned in the section "Scheduling Method" on page 119, PROC CPM assumes that activities cannot be preempted once they have started. Thus, an activity is scheduled only if it can be assured of enough resources throughout its entire duration. Sometimes, you may be able to make better use of the resources by allowing activities to be *split*. PROC CPM enables you to specify the maximum number of segments that an activity can be split into as well as the minimum duration of any segment of the activity. Suppose that for a given activity, d is its duration, $maxn$ is the maximum number of segments allowed, and $dmin$ is the minimum duration allowed for a segment. If one or the other of these values is not given, it is calculated appropriately based on the duration of the activity.

The scheduling algorithm described earlier is modified as follows:

- In Step 2, the procedure tries to schedule the entire activity (call it A) if it is critical. Otherwise, PROC CPM schedules, if possible, only the first part (say A1) of the activity (of length $dmin$). The remainder of the activity (call it A2, of length $d - dmin$) is added to the waiting list to be scheduled later. When it is A2's turn to be scheduled, it is again a candidate for splitting if the values of $maxn$ and $dmin$ allow it, and if it is not critical. This process is repeated until the entire activity has been scheduled.

- While ordering the activities in the waiting list, in case of a tie, the split segments of an activity are given priority over unsplit activities. Note that some scheduling rules could lead to more splitting than others.

- Activities that have an alignment type of MS or MF imposed on them by the ALIGNTYPE variable are not split.

Note that splitting may not always reduce project completion time; it is designed to make better use of resources. In particular, if there are gaps in resource availability, it allows activities to be split and scheduled around the gaps, thus using the resources more efficiently.

If activity splitting is allowed, a new variable is included in the Schedule data set called SEGMT_NO (*segment number*). If splitting does occur, the Schedule data set has more observations than the Activity data set. Activities that are not split are treated as before, except that the value of the variable SEGMT_NO is set to missing. For split activities, the number of observations output is one more than the number of disjoint segments created.

The first observation corresponding to such an activity has SEGMT_NO set to missing, and the S_START and S_FINISH times are set to be equal to the start and finish times, respectively, of the entire activity. That is, S_START is equal to the scheduled start time of the first segment, and S_FINISH is equal to the scheduled finish time of the last segment that the activity is split into. Following this observation, there are as many observations as the number of disjoint segments in the activity. All values for these additional observations are the same as the corresponding values for the first observation for this activity, except for the variables SEGMT_NO, S_START, S_FINISH, and the DURATION variable. SEGMT_NO is the index of the segment, S_START and S_FINISH are the resource-constrained start and finish times for this segment, and DURATION is the duration of this segment.

Actual Dates and Resource Allocation

The resource-constrained scheduling algorithm uses the early start schedule as the base schedule to determine possible start times for activities in the project. If an ACTUAL statement is used in the invocation of PROC CPM, the early start schedule (as well as the late start schedule) reflects the progress information that is specified for activities in the project, and thus affects the resource constrained schedule also. Further, activities that are already completed or in progress are scheduled at their actual start without regard to resource constraints. If the resource usage profile for such activities indicates that the resources are insufficient, a warning is printed to the log, but the activities are not postponed beyond their actual start time. The Usage data set contains negative values for the availability of the insufficient resources. These extra amounts are assumed to have come from the supplementary levels of the resources (if such a reservoir existed); for details about supplementary resources, see the section "Secondary Levels of Resources" on page 121.

If activity splitting is allowed (through the specification of the MINSEGMTDUR or MAXNSEGMT variable or the SPLITFLAG or TIMENOWSPLT option), activities that are currently in progress may be split at TIMENOW if resources are insufficient; then the second segment of the split activity is added to the list of activities that need to be scheduled subject to resource constraints. Starting from TIMENOW, all activities that are still unscheduled are treated as described in the section "Scheduling Method" on page 119.

Specifying Alternate Resources

PROC CPM enables you to identify alternate resources that can be substituted for any given resource that is insufficient. Thus, for example, you can specify that if programmer John is unavailable for a given task, he can be substituted by programmer David or Robert. This information is passed to PROC CPM through the Resource data set.

As with other aspects of the Resource data set, each observation is identified by a keyword indicating the type of information in that observation. Two keywords, ALTRATE and ALTPRTY, enable you to specify the rate of substitution and a prioritization of the alternate resources when a resource has more than one substitution (lower value indicates higher priority). Further, a new variable (identified to PROC CPM through the RESID= option) is used to identify the resource for which alternates are being specified in the current observation. Consider the following Resource data:

```
OBS  OBSTYPE   RES_NAME  RES_DATE   JOHN   DAVID   ROBERT

 1   RESTYPE                  .       1     1.0      1.0
 2   ALTRATE     JOHN         .       1     0.5      0.5
 3   ALTPRTY     JOHN         .       1     2.0      3.0
 4   RESLEVEL            15JUL04      1     1.0      1.0
```

In these Resource data, the second observation indicates that John can be substituted by David or Robert; however, either David or Robert can accomplish John's tasks with half the effort. In other words, if an activity requires 1 unit of John, it can also be accomplished with 0.5 units of David. Also, the third observation, with OBSTYPE='ALTPRTY', indicates that if John is unavailable, PROC CPM should first try to use David and if he, too, is unavailable, then should use Robert. This set up enables a wide range of control for specifying alternate resources.

In other words, the mechanism for specifying alternate resources is as follows: for each resource, specify a list of possible alternatives along with a conversion rate and an order in which the alternatives are to be considered. In the Resource data set, add another variable (identified by the RESID= option) to specify the name of the resource variable for which alternatives are being specified (the variable RES_NAME in the preceding example).

Let OBSTYPE='ALTRATE' for the observation that specifies the rate of conversion for each possible alternate resource (missing implies the particular resource cannot be substituted). For resources that *drive* an activity's duration, the specification of the alternate rate is used as a multiplier of the resource-driven duration. See the section "Resource-Driven Durations and Alternate Resources" on page 126 for details.

Let OBSTYPE='ALTPRTY' for the observation that specifies a prioritization for the resources.

All substitute resources must be of the same type (replenishable or consumable) as the primary resource. The specification of the RESID= option triggers the use of alternate resources. If alternate resources are used, the Schedule data set contains new variables that specify the actual resources that are used; the names of these variables are obtained by prefixing the resource names by 'U'. When activities are allowed to be split and alternate resources are allowed, different segments of the activity can use a different set of resources. If this is the case, the Schedule data set contains a different observation for every segment that uses a different set of resources, even if these segments are contiguous in time. Contiguous segments, even if they use different sets of resources, are not treated as true splits for the purpose of counting the number of splits allowed for the activity.

By default, multiple resources cannot be used to substitute for a single resource. To enable multiple alternates, use the MULTIPLEALTERNATES option or add an observation to the Resource data set identifying which resources allow multiple alternates. For details, see the section "Specifying Multiple Alternates" on page 125.

See Example 4.20 for an illustration of the use of alternate resources.

Specifying Multiple Alternates

As described in the section "Specifying Alternate Resources" on page 123, you can use the Resource data set to specify alternate resources for any given resource. You can specify a rate of substitution and a priority for substitution. However, the CPM procedure will not use multiple alternate resources to substitute for a given resource. For example, suppose that an activity needs two programmers and the available programmers (alternate resources) are John and Mary. By default, the CPM procedure cannot assign both John and Mary to the activity to fulfill the resource requirement of two programmers.

However, this type of substitution is useful to effectively model group resources or skill pools. To enable substitution of multiple alternates for a single resource, use the MULTIPLEALTERNATES option in the RESOURCE statement. This option enables all resources that have alternate specifications (through observations of the type ALTRATE or ALTPRTY in the Resource data set) to use multiple alternates. See Table 4.8 for details about type identifier variables.

You can refine this feature to selectively allow multiple substitution or set a minimum rate of substitution, by adding special observations to the Resource data set. As with other aspects of the Resource data set, the specifications related to multiple alternates are identified by observations with special keywords, MULTALT and MINARATE.

Let OBSTYPE='MULTALT' for the observation that identifies which resources can have multiple alternates. Valid values for such an observation are '0' and '1': '0' indicates that the resource cannot be substituted by multiple resources, and '1' indicates that it can be substituted by multiple resources. If the Resource data set contains such an observation, the MULTIPLEALTERNATES option is ignored and the values specified in the observation are used to allow multiple substitutions for only selected resources. See Table 4.8 for details about type identifier variables.

Let OBSTYPE='MINARATE' for the observation that indicates the minimum rate of substitution for each resource. For example, you may not want a primary resource requirement of 1.5 programmers, to be satisfied by 5 different alternate programmers at a rate of 0.3 each. To ensure that the minimum rate of substitution is 0.5, specify the value for the resource variable, PROGRAMMER, as '0.5' in the observation with OBSTYPE='MINARATE'. In other words, use this observation if you do not wish to split an activity's resource requirement across several alternate resources with a very small rate of utilization per resource. See Table 4.8 for details about type identifier variables. Consider the following Resource data:

```
OBS   OBSTYPE    RES_NAME  RES_DATE    JOHN    DAVID    ROBERT

 1    RESTYPE                     .       1        1         1
 2    ALTRATE      JOHN           .       1        2         2
 3    MULTALT       .             .       1        .         .
 4    MINARATE      .             .     0.5        .         .
 5    RESLEVEL            15JUL04         0      1.0       1.0
```

In these Resource data, observations 3 and 4 control the use of multiple alternates. They specify that a requirement for John can be substituted with multiple alternates. Further, if multiple alternates are used instead of John, do not allocate them in units less than 0.5. Observation 2 indicates that David and Robert require twice the effort to accomplish John's tasks. Thus, if an activity requires 1 unit of John, and he is unavailable, the CPM procedure will require 2 units of David (or Robert) to substitute for John. However, only 1 unit each of David and Robert is available. If multiple alternates are *not* allowed, the resource allocation algorithm will fail. However, since the resource John *does* allow multiple substitution, the activity can be scheduled with 1 unit of David and 1 unit of Robert (each substituting for 1/2 of the requirement for John).

Allowing multiple alternates for a single resource raises an interesting question: When distributing the resource requirements across multiple alternatives, should the primary resource be included in the list of multiple alternates? For instance, in the preceding example, if the resource level for John is ‘0.5’ (in observation 5), should the activity use John at rate 0.5 and assign the remainder to one (or more) of the alternate resources? Or, should the primary resource be excluded from the list of possible alternates? You can select either behavior for the primary resource by specifying ‘1’ (for inclusion) or ‘0’ (for exclusion) in the observation with OBSTYPE=‘ALTRATE’ that corresponds to the primary resource (with RES_NAME=‘JOHN’). Thus, in the preceding example, John can be one of the multiple alternates when substituting for himself. To exclude John from the list, set the value of the variable JOHN to ‘0’ in observation 2. You will also need to set the value of JOHN to ‘0’ in any observation with OBSTYPE=‘ALTPRTY’ and RES_NAME=‘JOHN’.

Resource-Driven Durations and Alternate Resources

the section “Specifying Alternate Resources” on page 123 describes the use of the RESID= option and the observations of type ‘ALTRATE’ and ‘ALTPRTY’ in the Resource data set to control the use of alternate resources during resource allocation. The behavior described in that section refers to the substitution of resources for resources that have a fixed duration. Alternate resources can also be specified for resources that drive an activity’s duration. However, the specification of the alternate rate is interpreted differently: it is used as a multiplier of the resource-driven duration.

For example, consider the following Resource data:

```
OBS   OBSTYPE    RES_NAME  RES_DATE    JOHN   DAVID   ROBERT

 1    RESTYPE       .           .        1      1        1
 2    RESRCDUR      .           .        1      1        1
 3    ALTRATE     JOHN          .        1      2        2
 4    ALTPRTY     JOHN          .        1      2        3
 5    RESLEVEL      .       15JUL04      .     1.0      1.0
```

In these Resource data, the second observation indicates that all the resources are driving resources. The third observation indicates that John can be substituted by David or Robert; however, either David or Robert will require twice as long to accomplish John’s tasks for resource-driven activities. Thus, in contrast to the fixed-duration activities, the ALTRATE specification changes the *duration* of the alternate resource, not the *rate of use*.

For instance, consider the following activity with the specified values for the DURATION and WORK variables and the resource requirement for John:

```
OBS   ACTIVITY  DURATION  WORK  JOHN   DAVID   ROBERT

 1    Act1          3      10     1      .       .
```

Activity ‘Act1’ requires 10 days of work from John, indicating that the resource-driven duration for Act1 is 10 days. However, from the preceding Resource data, John is not available, but can be substituted by David or Robert, who will require twice as long to accomplish the work. So, if Act1 is scheduled using either one of the alternate resources, its resource-driven duration will be 20 days.

Auxiliary Resources

Sometimes, the use of a certain resource may require simultaneous use of other resources. For example, use of a crane will necessitate the use of a crane operator. In other words, if an activity needs the resource, CRANE, it will also need a corresponding resource, CRANEOP. Such requirements can be easily modeled by adding both CRANE and CRANEOP to the list of resources required by the activity.

However, when alternate resources are used, the problem becomes more complex. For example, suppose an activity requires a CRANE and there are two possible cranes that can be used, CRANE1 and CRANE2. You can specify CRANE1 and CRANE2 as the alternate resources for CRANE. Suppose further that each of the two cranes has a specific operator, CRANEOP1 and CRANEOP2, respectively. Specifying CRANEOP1 and CRANEOP2 separately as alternates for CRANEOP will not necessarily guarantee that CRANEOP1 (or CRANEOP2) is used as the alternate for CRANEOP in conjunction with the use of the corresponding CRANE1 (or CRANE2).

You can model such a situation by the use of Auxiliary resource specification: specify CRANEOP1 and CRANEOP2 as auxiliary resources for CRANE1 and CRANE2, respectively. Auxiliary resources are specified through the Resource data set, using observations identified by the keyword AUXRES for the value of the OBSTYPE variable. For an observation of this type, the RESID variable specifies the name of the primary resource. (This is similar to the specification of ALTRATE and ALTPRTY.) See Table 4.8 for details about type identifier variables.

Once auxiliary resources are specified in the Resource data set, it is sufficient to specify only the primary resource requirements in the Activity data set. In this situation, for example, it is sufficient to require a CRANE for the activity in the Activity data set.

In the Resource data set, add a new observation type, ‘AUXRES’, which will specify the auxiliary resources that are needed for each primary resource. For an observation of this type, the RESID variable specifies the name of the primary resource. The value for each auxiliary resource indicates the rate at which it is required whenever the primary resource is used. You will also need to specify CRANE1 and CRANE2 as the alternate resources for CRANE in the Resource data set.

When scheduling the activity, PROC CPM will schedule CRANE1 (or CRANE2) as the alternate only if both CRANE1 and CRANEOP1 (or CRANE2 and CRANEOP2) are available.

For instance, the preceding example will have the following Resource data set:

```
OBSTYPE    RESID    PER      CRANE  CRANE1  CRANE2  CRANEOP1  CRANEOP2

AUXRES     CRANE1    .         .      .       .        1         .
AUXRES     CRANE2    .         .      .       .        .         1
ALTRATE    CRANE     .         .      1       1        .         .
RESLEVEL     .      10JUL04    .      1       1        1         1
```

RESOURCEOUT= Usage Data Set

The RESOURCEOUT= data set (referred to as the Usage data set) contains information about the resource usage for the resources specified in the RESOURCE statement. The options ALL, AVPROFILE, ESPROFILE, LSPROFILE, and RCPROFILE (each is discussed earlier in the section “RESOURCE Statement” on page 80) control the number of variables that are to be created in this data set. The ROUTINTERVAL= and ROUTINTPER= options control the number of observations that this data set is to contain. Of the options controlling the number of variables, AVPROFILE and RCPROFILE are allowed only if the procedure is used to obtain a resource-constrained schedule.

The Usage data set always contains a variable named _TIME_ that specifies the date for which the resource usage or availability in the observation is valid. For each of the variables specified in the RESOURCE statement, one, two, three, or four new variables are created depending on how many of the four possible options (AVPROFILE, ESPROFILE, LSPROFILE, and RCPROFILE) are in effect. If none of these four options is specified, the ALL option is assumed to be in effect. Recall that the ALL option is equivalent to specifying ESPROFILE and LSPROFILE when PROC CPM is used to obtain an unconstrained schedule, and it is equivalent to specifying all four options when PROC CPM is used to obtain a resource-constrained schedule.

The new variables are named according to the following convention:

- The prefix A is used for the variable describing the resource availability profile.
- The prefix E is used for the variable denoting the early start usage.
- The prefix L is used for the variable denoting the late start usage.
- The prefix R is used for the variable denoting the resource-constrained usage.

The suffix is the name of the resource variable if the name is less than the maximum possible variable length (which is dependent on the VALIDVARNAME option). If the length of the name is equal to this maximum length, the suffix is formed by deleting the character following the *(n/2)th* position. The user must ensure that this naming convention results in unique variable names in the Usage data set.

The ROUTINTERVAL=*routinterval* and ROUTINTPER=*routintper* options specify that two successive values of the _TIME_ variable differ by *routintper* number of *routinterval* units, measured with respect to a specific calendar. If the *routinterval* is not specified, PROC CPM selects a default value depending on the format of the start and finish variables in the Schedule data set. The value of *routinterval* is indicated in a message written to the SAS log.

The MINDATE=*mindate* and MAXDATE=*maxdate* options specify the minimum and maximum values of the _TIME_ variable, respectively. Thus, the Usage data set has observations containing the resource usage information from *mindate* to *maxdate* with the time interval between the values of the _TIME_ variable in two successive observations being equal to *routintper* units of *routinterval*, measured with respect to a specific calendar. For example, if *routinterval* is MONTH and *routintper* is 3, then the time interval between successive observations in the Usage data set is three months.

The calendar used for incrementing the _TIME_ variable is specified using the AROUTCAL= or NROUTCAL= options depending on whether the calendars for the project are specified using alphanumeric or numeric values, respectively. In the absence of either of these specifications, the default calendar is used. For example, if the default calendar follows a five-day work week and ROUTINTERVAL=DAY, the Usage data set will not contain observations corresponding to Saturdays and Sundays. You can also use the ROUTNOBREAK option to indicate that there should be no breaks in the _TIME_ values due to breaks or holidays.

Interpretation of Variables

The availability profile indicates the amount of resources available at the beginning of the time interval specified in the _TIME_ variable, after accounting for the resources used through the previous time period.

By default, each observation in the Resource Usage data set indicates the *rate* of resource usage per unit *routinterval* at the start of the time interval specified in the _TIME_ variable. Note that *replenishable resources* are assumed to be tied to an activity during any of the activity's breaks or holidays that fall in the course of the activity's duration. For *consumable resources*, you can use the CUMUSAGE option to obtain *cumulative usage* of the resource, instead of *daily rate of usage*. Often, it is more useful to obtain *cumulative usage* for consumable resources.

You can use the TOTUSAGE option on the RESOURCE statement to get the *total* resource usage for each resource within each time period. If you wish to obtain both the *rate* of usage and the *total* usage for each time period, use the APPEND option on the RESOURCE statement.

The following example illustrates the default interpretation of the new variables.

Suppose that for the data given earlier (see the section "Specifying Resource Requirements" on page 118), activities 'A' and 'B' have S_START equal to 1JUL04 and 5JUL04, respectively. If the RESOURCE statement has the options AVPROFILE and RCPROFILE, the Usage data set has these five variables, _TIME_, RWORKERS, AWORKERS, RBRICKS, and ABRICKS. Suppose further that *routinterval* is DAY and *routintper* is 1. The Usage data set contains the following observations:

```
_TIME_      RWORKERS   AWORKERS    RBRICKS    ABRICKS

1JUL04         0           0          100        1000
2JUL04         0           0          100         900
3JUL04         0           0          100         800
4JUL04         0           0          100         700
5JUL04         2           2          100         600
6JUL04         2           2            0         500
7JUL04         2           2            0         500
8JUL04         2           2            0         500
9JUL04         0           4            0        1000
```

On each day of activity A's duration, the resource BRICKS is consumed at the rate of 100 bricks per day. At the beginning of the first day (July 1, 2004), all 1000 bricks are still available. Each day the availability drops by 100 bricks, which is the rate of consumption. On July 5, activity 'B' is scheduled to start. On the four days starting with July 5, the value of RWORKERS is '2', indicating that 2 workers are used on each of those days leaving an available supply of 2 workers (AWORKERS is equal to '2' on all 4 days).

If ROUTINTPER is set to 2, and the CUMUSAGE option is used, then the observations would be as follows:

```
_TIME_      RWORKERS   AWORKERS    RBRICKS    ABRICKS

1JUL04         0           0            0        1000
3JUL04         0           0          200         800
5JUL04         2           2          400         600
7JUL04         2           2          500         500
9JUL04         0           4          500        1000
```

The value of RBRICKS indicates the *cumulative* usage of the resource BRICKS through the *beginning* of the date specified by the value of the variable _TIME_ in each observation. That is why, for example, RBRICKS = 0 on 1JUL04 and not 200.

If the procedure uses supplementary levels of resources, then, on a day when supplementary levels of resources were used through the beginning of the day, the value for the availability profile for the relevant resources would be negative. The absolute magnitude of this value would denote the amount of supplementary resource that was used through the beginning of the day. For instance, if ABRICKS is '−100' on 11JUL04, it would indicate that 100 bricks from the supplementary reservoir were used through the end of July 10, 2004. See Example 4.16, "Using Supplementary Resources," and Example 4.17, "INFEASDIAGNOSTIC Option and Aggregate Resource Type."

If, for the same data, ROUTINTPER is 2, and the APPEND option is specified, the Usage data set would contain two sets of observations, the first indicating the *rate of resource usage per day*, and the second set indicating the *product of the rate and the time interval between two successive observations*. The observations (five in each set) would be as follows:

```
 _TIME_     OBS_TYPE    RWORKERS    RBRICKS

01JUL04     RES_RATE        0          100
03JUL04     RES_RATE        0          100
05JUL04     RES_RATE        2          100
07JUL04     RES_RATE        2            0
09JUL04     RES_RATE        0            0
01JUL04     RES_USED        0          200
03JUL04     RES_USED        0          200
05JUL04     RES_USED        4          100
07JUL04     RES_USED        4            0
09JUL04     RES_USED        0            0
```

Variable Usage Profile for Consumable Resources

For consumable resources that have a variable usage profile (as indicated by the values 1 or 2 for observations of type RESUSAGE in the Resource data set), the values of the usage variables indicate the amount of the resource consumed by an activity at the beginning or end of the activity. For example, consider the resources PAYMENT and ADVANCE specified in the following Resource data set:

```
OBS   OBSTYPE     DATE       WORKERS    BRICKS    PAYMENT  ADVANCE

 1    RESTYPE       .           1          2         2        2
 2    RESUSAGE      .           .          .         2        1
 3    RESLEVEL   1JUL2004       4        1000      2000      500
```

Suppose the activity 'Task 1', specified in the following observation, is scheduled to start on July 1, 2004:

```
OBS   ACTIVITY    DUR      WORKERS     BRICKS  PAYMENT  ADVANCE

 1     Task 1      5         1          100     1000      200
```

For these data, the resource usage profile for the resources will be as indicated in the following output:

```
_TIME_      RWORKERS    RBRICKS    RPAYMENT    RADVANCE

1JUL04         1          100          0          200
2JUL04         1          100          0            0
3JUL04         1          100          0            0
4JUL04         1          100          0            0
5JUL04         1          100          0            0
6JUL04         0            0       1000            0
```

RESOURCESCHED= Resource Schedule Data Set

The Resource Schedule data set (requested by the RESSCHED= option on the CPM statement) is very similar to the Schedule data set, and it contains the start and finish times for each resource used by each activity. The data set contains the variables listed in the ACTIVITY, TAILNODE, and HEADNODE statements and all the relevant schedule variables (E_START, E_FINISH, and so forth). For each activity in the project, this data set contains the schedule for the entire activity as well as the schedule for each resource used by the activity. The variable RESOURCE identifies the name of the resource to which the observation refers; the value of the RESOURCE variable is missing for observations that refer to the entire activity's schedule. The variable DUR_TYPE indicates whether the resource is a driving resource or a spanning resource or whether it is of the fixed type.

A variable _DUR_ indicates the duration of the activity for the resource identified in that observation. This variable has missing values for resources that are of the spanning type. For resources that are of the driving type, the variable _WORK_ shows the total amount of work required by the resource for the activity in that observation. The variable R_RATE shows the rate of usage of the resource for the relevant activity. For driving resources, the variable _DUR_ is computed as (WORK / R_RATE).

If you specify an ACTUAL statement, the Resource Schedule data set also contains the STATUS variable indicating whether the resource has completed work on the activity, is in progress, or is still pending.

Multiproject Scheduling

The CPM procedure enables you to define activities in a multiproject environment with multiple levels of nesting. You can specify a PROJECT variable that identifies the name or number of the project to which each activity belongs. The PROJECT variable must be of the same type and length as the ACTIVITY variable. Further, each project can be considered as an activity, enabling you to specify precedence constraints, alignment dates, or progress information for the different projects. Precedence constraints can be specified between two projects, between activities in the same or different projects, or between a project and activities in another project.

The PROJECT variable enables you to specify the name of the project to which each activity belongs. Each project can in turn be treated as an activity that belongs to a bigger project. Thus, the (PROJECT, ACTIVITY) pair of variables enables you to specify multiple levels of nesting using a hierarchical structure for the (task, supertask) relationship.

In the following discussion, the terms superproject, supertask, parent task, ancestor task, project, or subproject refer to a *composite* task (a task composed of other tasks). A lowest level task (one which has no subtasks under it) is referred to as a child task, descendent task, a *leaf* task, or a *regular* task.

You can assign most of the "activity attributes" to a supertask; however, some of the interpretations may be different. The significant differences are listed as follows.

Activity Duration

Even though a supertask has a value specified for the DURATION variable, the finish time of the supertask may not necessarily be equal to the (start time + duration). The start and finish times of a parent task (supertask) always encompass the span of all its subtasks. In other words, the start (finish) time of a supertask is the minimum start (maximum finish) time of all its subtasks.

The specified DURATION for a supertask is used only if the USEPROJDUR option is specified; this variable is used to compute an upper bound on the late finish time of the project. In other words, you can consider the duration of a supertask as a *desired* duration that puts a constraint on its finish time.

NOTE: You cannot specify resource-driven durations for supertasks.

Precedence Constraints

You cannot specify a Start-to-Finish or Finish-to-Finish type of precedence constraint when the Successor task is a supertask. Such a constraint is ignored, and a warning is written to the log.

Time Constraints

The CPM procedure supports all the customary time constraints for a supertask. However, since the supertask does not really have an inherent duration, some of the constraints may lead to unexpected results.

For example, a constraint of the type SLE (Start Less than or Equal to) on a leaf task uses the task's duration to impose a maximum late finish time for the task. However, for a supertask, the duration is determined by the span of all its subtasks, which may depend on the activities' calendars. The CPM procedure uses an estimate of the supertask's duration computed on the basis of the precedence constraints to determine the maximum finish time for the supertask using the date specified for the SLE constraint. Such a constraint may not translate to the correct upper bound on the supertask's finish time if the project has multiple calendars. The presence of multiple calendars could change the computed duration of the supertask depending on the starting date of the supertask. Thus, in general, it is better to specify SGE (Start Greater than or Equal to) or FLE (Finish Less than or Equal to) constraints on supertasks.

Note that alignment constraints of the type SGE or FLE percolate down the project hierarchy. For example, if there is an SGE specification on a supertask, then all the subtasks of this supertask must also start on or after the specified date.

Mandatory constraints (either of the type MS or MF) are used to set fixed start and finish times on the relevant task. Such constraints are checked for consistency between a parent task and all its descendants.

Progress Information

You can enter progress information for supertasks in the same way as you do for leaf tasks. The procedure attempts to reconcile inconsistencies between the actual start and finish times of a parent and its children. However, it is sufficient (and less ambiguous) to enter progress information only about the tasks at the lowest level.

Resource Requirements

You can specify resource requirements for supertasks in the same way as you do for regular tasks. However, the supertask is scheduled in conjunction with all its subtasks. In other words, a leaf task is scheduled only when *its resources and the resources for all its ancestors* are available in sufficient quantity. Thus, a supertask needs to have enough resources throughout the schedule of any of its subtasks; in fact, the supertask needs to have enough resources throughout its entire span. In other words, a supertask's resource requirements are treated as "spanning."

In addition to the above treatment of a supertask's resources, there are two other resource scheduling options available for handling the resource requirements of supertasks. You can use the AGGRE-

GATEPARENTRES option in the PROJECT statement to indicate that a supertask's resource requirements are to be used only for aggregation. In other words, resource allocation is performed taking into account the resource requirements of only the leaf tasks. Alternately, you can select to ignore any resource requirements specified for supertasks by specifying the IGNOREPARENTRES option. Note the difference between the AGGREGATEPARENTRES and IGNOREPARENTRES options. The first option includes the supertask's requirements while computing the aggregate resource usage, while the second option is equivalent to setting all parent resource requirements to 0.

Resource-Driven Durations

Any WORK specification is ignored for a parent task. Resources required for a supertask cannot drive the duration of the task; a supertask's duration is driven by all its subtasks. Note that each leaf task can still be resource driven.

Schedule Computation

The project hierarchy and all the precedence constraints (between leaf tasks, between supertasks, or between a supertask and a leaf task) are taken into consideration when the project schedule is computed. A task (parent or leaf) can be scheduled only when *its precedences and all its parent's precedences* are satisfied.

During the forward pass of the scheduling algorithm, all independent start tasks (leaf tasks or supertasks with no predecessors) are initialized to the project start date. Once a supertask's precedences (if any) are satisfied, all its subtasks whose precedences have been satisfied are added to the list of activities that can be scheduled. The early start times for the subtasks are initialized to the early start time of the supertask and are then updated, taking into account the precedence constraints and any alignment constraints on the activities.

Once all the subtasks are scheduled, a supertask's early start and finish times are set to the minimum early start and maximum early finish, respectively, of all its subtasks.

The late start schedule is computed using a backward pass through the project network, considering the activities in a reverse order from the forward pass. The late schedule is computed starting with the last activity (activities) in the project; the late finish time for each such activity is set to the master project's finish date. By default, the master project's finish date is the maximum of the early finish dates of all the activities in the master project (if a FINISHBEFORE date is specified with the FBDATE option, this date is used as the starting point for the backward calculations).

During the backward pass, the late finish time of a supertask is determined by the precedence constraints and any alignment specification on the supertask. You can specify a finish constraint on a supertask by using the ALIGNDATE and ALIGNTYPE variables, or by using the SEPCRIT or USEPROJDUR option.

If a finish constraint is specified using the ALIGNDATE and ALIGNTYPE specifications, the L_FINISH for the supertask is initialized to this value. If the SEPCRIT option is specified, the supertask's late finish time is initialized to its early finish time. If the USEPROJDUR option is specified, the late finish time for the supertask is initialized using the early start time of the supertask and the specified supertask duration. The late finish time of the supertask could further be affected by the precedence constraints. Once a supertask's late finish has been determined, this value is treated as an upper bound on the late finish of all its subtasks.

As with the early start schedule, once all the subtasks have been scheduled, the late start and finish times for a supertask are set to the minimum late start and maximum late finish time, respectively, of all its subtasks.

Schedule Data Set

If a PROJECT variable is specified, the Schedule data set contains the PROJECT variable as well as two new variables called PROJ_DUR and PROJ_LEV.

The PROJ_DUR variable contains the project duration (computed as E_FINISH - E_START of the project) for each superproject in the master project. This variable has missing values for the leaf tasks. It is possible for (L_FINISH - L_START) to be different from the value of PROJ_DUR. If a resource-constrained schedule is produced by PROC CPM, the project duration is computed using the resource constrained start and finish times of the superproject; in other words, in this case PROJ_DUR = (S_FINISH - S_START).

The PROJ_LEV variable specifies the depth of each activity from the root of the project hierarchy tree. The root of the tree has PROJ_LEV = 0; If the project does not have a single root, a common root is defined by the CPM procedure.

The ADDACT option on the PROC CPM statement causes an observation to be added to the Schedule data set for this common root. This observation contains the project start and finish times and the project duration. The ADDACT option also adds an observation for any activity that may appear as a value of the SUCCESSOR or PROJECT variable without appearing as a value of the ACTIVITY variable.

In addition to the PROJ_DUR and PROJ_LEV variables, you can request that a WBS code be added to the output data set (using the option ADDWBS). You can also add variables, ES_ASC, ES_DESC, LS_ASC, LS_DESC, SS_ASC, and SS_DESC, that indicate a sorting order for activities in the output data set. For example, the variable ES_ASC enables you to sort the output data set in such a way that the activities within each superproject are ordered according to increasing early start time.

Macro Variable _ORCPM_

The CPM procedure defines a macro variable named _ORCPM_ . This variable contains a character string that indicates the status of the procedure. It is set at procedure termination. The form of the _ORCPM_ character string is STATUS=, optionally followed by REASON=. The value for STATUS= is one of the following:

- SUCCESSFUL - indicates successful completion
- ERROR_EXIT - indicates unsuccessful termination

If PROC CPM terminates unsuccessfully, REASON= will be followed by one of the values below:

- CYCLE - indicates that PROC CPM found a cycle in the project network; activity/successor (or tailnode/headnode for an AOA representation) and parent/child (see PROJECT statement) pairs define precedence relationships among the activities, which can be viewed as directed arcs in a graph
- RES_INFEASIBLE - indicates that PROC CPM could not find a schedule that satisfies the resource requirements, given the specified resource levels; the INFEASDIAGNOSTIC option might be useful in this case
- BADDATA_ERROR - points to problems with the input specification; an example would be neglecting to name an activity or duration variable

- MEMORY_ERROR - occurs when memory is exhausted
- IO_ERROR - triggered by problems with a data set
- SEMANTIC_ERROR - indicates an incongruity in the user specification or input data
- SYNTAX_ERROR - occurs when the user has specified an invalid value for a keyword, for instance
- CPM_BUG - should not occur
- UNKNOWN_ERROR - should not occur

This information can be used when PROC CPM is one step in a larger program that needs to determine whether the procedure terminated successfully or not. Because _ORCPM_ is a standard SAS macro variable, it can be used in the ways that all macro variables can be used.

Input Data Sets and Related Variables

The CPM procedure uses activity, resource, and holiday data from several different data sets with key variable names being used to identify the appropriate information. Table 4.9 lists all of the variables associated with each input data set and their interpretation by the CPM procedure. The variables are grouped according to the statement that they are identified in. Some variables use default names and are not required to be identified in any statement.

Table 4.9 PROC CPM Input Data Sets and Associated Variables

Data Set	Statement	Variable Name	Interpretation
CALEDATA	CALID	CALID	Calendar corresponding to work pattern
	Default names	D_LENGTH	Length of standard work day
		SUN . . . _SAT_	Work pattern on day of week, valid values: WORKDAY, HOLIDAY, or one of the numeric variables in the Workday data set
DATA	ACTIVITY	ACTIVITY	Activity in AON format
	ACTUAL	A_START	Actual start time of activity
		A_FINISH	Actual finish time of activity
		REMDUR	Remaining duration
		PCTCOMP	Percentage of work completed
	ALIGNDATE	ALIGNDATE	Time constraint on activity

Table 4.9 (continued)

Data Set	Statement	Variable Name	Interpretation
	ALIGNTYPE	ALIGNTYPE	Type of time constraint, valid values: SGE, SEQ, SLE, FGE, FEQ, FLE, MS, MF
	BASELINE	B_START	Baseline start time of activity
		B_FINISH	Baseline finish time of activity
	CALID	CALID	Calendar followed by activity
	DURATION	DURATION	Duration of activity
		FINISH	Finish time of activity
		START	Start time of activity
	HEADNODE	HEADNODE	Head of arrow (arc) in AOA format
	ID	ID	Additional project information
	PROJECT	PROJECT	Project to which activity belongs
	RESOURCE	ACTDELAY	Activity delay
		ACTPRTY	Activity priority
		MAXNSEGMT	Maximum number of segments
		MINSEGMTDUR	Minimum duration of a segment
		RESOURCE	Amount of resource required
		WORK	Amount of work required
	SUCCESSOR	SUCCESSOR	Successor in AON format
		LAG	Nonstandard precedence relationship
	TAILNODE	TAILNODE	Tail of arrow (arc) in AOA format

Table 4.9 (continued)

Data Set	Statement	Variable Name	Interpretation
HOLIDATA	CALID	CALID	Calendar to which holiday applies
	HOLIDAY	HOLIDAY	Start of holiday
		HOLIDUR	Duration of holiday
		HOLIFIN	End of holiday
RESOURCEIN	RESOURCE	OBSTYPE	Type of observation; valid values: RESLEVEL, RESTYPE, SUPLEVEL, RESPRTY, ALTRATE, ALTPRTY, RESUSAGE, AUXRES, MULTALT, MINARATE, CALENDAR
		PERIOD	Time from which resource is available
		RESID	Resource for which alternates are given
		RESOURCE	Resource type, priority, availability, alternate rate, alternate priority
WORKDATA		Any numeric variable	On-off pattern of work (shift definition)

Missing Values in Input Data Sets

The following table summarizes the treatment of missing values for variables in the input data sets used by PROC CPM.

Table 4.10 Treatment of Missing Values in the CPM Procedure

Data Set	Variable	Value Used / Assumption Made / Action Taken
CALEDATA	CALID	Default calendar (0 or DEFAULT)
	D_LENGTH	DAYLENGTH, if available. 8:00, if INTERVAL = WORKDAY, DTWRKDAY 24:00, otherwise
	SUN . . . _SAT_	Corresponding shift for default calendar
DATA	ACTIVITY	Input error: procedure stops with error message
	ACTDELAY	DELAY= specification
	ACTPRTY	Infinity (indicates lowest priority)
	ALIGNDATE	Project start date for start activity

Table 4.10 (continued)

Data Set	Variable	Value Used / Assumption Made / Action Taken
	ALIGNTYPE	SGE: if ALIGNDATE is not missing
	A_FINISH	See the section "Progress Updating" on page 109 for details
	A_START	See the section "Progress Updating" on page 109 for details
	B_FINISH	Updated if UPDATE= option is on
	B_START	Updated if UPDATE= option is on
	CALID	Default calendar (0 or DEFAULT)
	DURATION	Input error: procedure stops with error message
	FINISH	Value ignored
	HEADNODE	Input error: procedure stops with error message
	ID	Missing
	LAG	FS_0: if corresponding successor Variable value is not missing
	MAXNSEGMT	Calculated from MINSEGMTDUR
	MINSEGMTDUR	0.2 * DURATION
	PCTCOMP	See the section "Progress Updating" on page 109 for details
	PROJECT	Activity is at highest level
	REMDUR	See the section "Progress Updating" on page 109 for details
	RESOURCE	0
	START	Value ignored
	SUCCESSOR	Value ignored
	TAILNODE	Input error: procedure stops with error message
	WORK	Resources use fixed duration
HOLIDATA	CALID	Holiday applies to all calendars defined
	HOLIDAY	Observation ignored
	HOLIDUR	Ignored if HOLIFIN is not missing; 1, otherwise
	HOLIFIN	Ignored if HOLIDUR is not missing; HOLIDAY + (1 unit of INTERVAL), otherwise
RESOURCEIN	OBSTYPE	RESLEVEL
	PERIOD	Input error if OBSTYPE is RESLEVEL, otherwise ignored
	RESID	Observation ignored
	RESOURCE	1.0, if OBSTYPE is RESTYPE infinity, if OBSTYPE is RESPRTY 0.0, if OBSTYPE is RESUSAGE 0.0, if OBSTYPE is SUPLEVEL 0.0, if OBSTYPE is RESLEVEL and this is the first observation of this type otherwise, equal to value in previous observation
WORKDATA	Any numeric variable	00:00, if first observation 24:00, otherwise

FORMAT Specification

As can be seen from the description of all of the statements and options used by PROC CPM, the procedure handles SAS date, time, and datetime values in several ways: as time constraints on the activities, holidays specified as date or datetime values, periods of resource availabilities, actual start and finish times, and several other options that control the scheduling of the activities in time. The procedure tries to reconcile any differences that may exist in the format specifications for the different variables. For example, if holidays are formatted as SAS date values while alignment constraints are specified in terms of SAS datetime values, PROC CPM converts all of the holidays to SAS datetime values suitably. However, the procedure needs to know how the variables are to be interpreted (as SAS date, datetime, or time values) in order for this reconciliation to be correct. Thus, it is important that you always use a FORMAT statement explicitly for each SAS date, time, or datetime variable that is used in the invocation of PROC CPM.

Computer Resource Requirements

There is no inherent limit on the size of the project that can be scheduled with the CPM procedure. The number of activities and precedences, as well as the number of resources are constrained only by the amount of memory available. Naturally, there needs to be a sufficient amount of core memory available in order to invoke and initialize the SAS system. As far as possible, the procedure attempts to store all the data in core memory.

However, if the problem is too large to fit in core memory, the procedure resorts to the use of utility data sets and swaps between core memory and utility data sets as necessary, unless the NOUTIL option is specified. The procedure uses the NACTS=, NADJ=, NNODES=, and NRESREQ= options to determine approximate problem size. If these options are not specified, the procedure estimates default values on the basis of the number of observations in the Activity data set. See the section "Syntax: CPM Procedure" on page 61 for default specifications.

The storage requirement for the data area required by the procedure is proportional to the number of activities and precedence constraints in the project and depends on the number of resources required by each activity. The time required depends heavily on the number of resources that are constrained and on how tightly constrained they are.

Examples: CPM Procedure

This section contains examples that illustrate several features of the CPM procedure. Most of the available options are used in at least one example. Two tables, Table 4.13 and Table 4.14, at the end of this section list all the examples in this chapter and the options and statements in the CPM procedure that are illustrated by each example.

A simple project concerning the manufacture of a widget is used in most of the examples in this section. Example 4.22 deals with a nonstandard application of PROC CPM and illustrates the richness of the modeling environment that is available with the SAS System. The last few examples use different projects to illustrate multiproject scheduling and resource-driven durations. resource calendars and negative resource requirements.

There are 14 activities in the widget manufacturing project. Example 4.1 and Example 4.2 illustrate a basic project network that is built upon by succeeding examples. The tasks in the project can be classified by the division or department that is responsible for them.

Table 4.11 lists the detailed names (and corresponding abbreviations) of all the activities in the project and the department that is responsible for each one. As in any typical project, some of these activities must be completed before others. For example, the activity 'Approve Plan' must be done before any of the activities 'Drawings', 'Study Market', and 'Write Specs', can start. Table 4.12 summarizes the relationships among the tasks and gives the duration in days to complete each task. This table shows the relationship among tasks by listing the immediate successors to each task.

Table 4.11 Widget Manufacture: Activity List

Task	Department	Activity Description
Approve Plan	Planning	Finalize and Approve Plan
Drawings	Engineering	Prepare Drawings
Study Market	Marketing	Analyze Potential Markets
Write Specs	Engineering	Write Specifications
Prototype	Engineering	Build Prototype
Mkt. Strat.	Marketing	Develop Marketing Concept
Materials	Manufacturing	Procure Raw Materials
Facility	Manufacturing	Prepare Manufacturing Facility
Init. Prod.	Manufacturing	Initial Production Run
Evaluate	Testing	Evaluate Product In-House
Test Market	Testing	Mail Product to Sample Market
Changes	Engineering	Engineering Changes
Production	Manufacturing	Begin Full Scale Production
Marketing	Marketing	Begin Full Scale Marketing

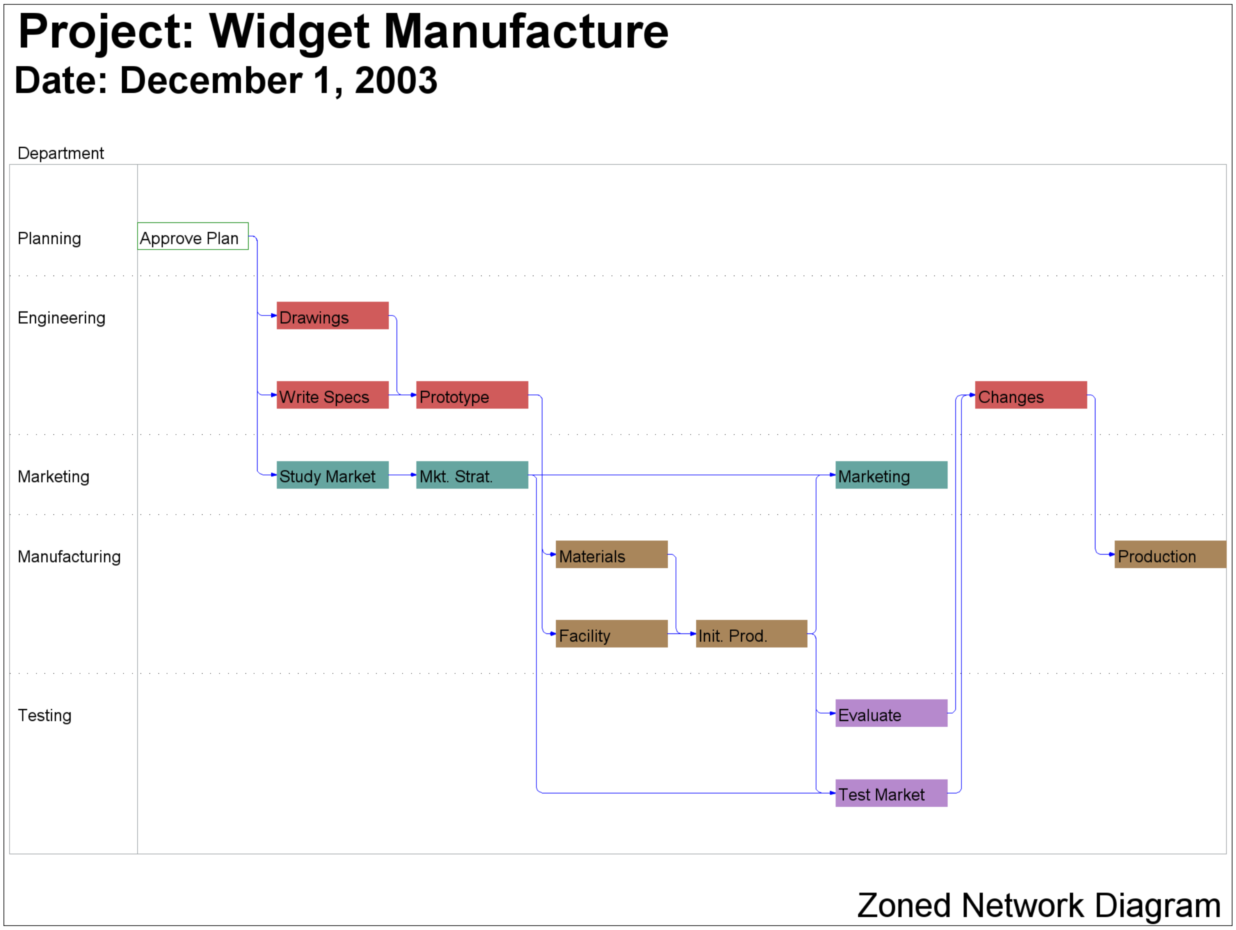
Table 4.12 Widget Manufacture: Precedence Information

Task	Dur	Successor	Successor	Successor
Approve Plan	10	Drawings	Study Market	Write Specs
Drawings	20	Prototype		
Study Market	10	Mkt. Strat.		
Write Specs	15	Prototype		
Prototype	30	Materials	Facility	
Mkt. Strat.	25	Test Market	Marketing	
Materials	60	Init. Prod.		
Facility	45	Init. Prod.		
Init. Prod.	30	Test Market	Marketing	Evaluate
Evaluate	40	Changes		
Test Market	30	Changes		
Changes	15	Production		
Production	0			
Marketing	0			

The relationship among the tasks can be represented by the network in Output 4.1.1. The diagram was produced by the NETDRAW procedure. The code used is the same as in Example 9.11 in Chapter 9, "The NETDRAW Procedure," although the colors may be different.

Example 4.1: Activity-on-Node Representation

Output 4.1.1 Network Showing Task Relationships in Activity-on-Node Format



The following DATA step reads the project network in AON format into a SAS data set named WIDGET. The data set contains the minimum amount of information needed to invoke PROC CPM, namely, the ACTIVITY variable, one or more SUCCESSOR variables, and a DURATION variable. PROC CPM is invoked, and the Schedule data set is displayed using the PRINT procedure in Output 4.1.2. The Schedule data set produced by PROC CPM contains the solution in canonical units, without reference to any calendar date or time. For instance, the early start time of the first activity in the project is the beginning of period 0 and the early finish time is the beginning of period 5.

```
/* Activity-on-Node representation of the project */
   data widget;
   format task $12. succ1-succ3 $12.;
   input task & days succ1 & succ2 & succ3 & ;
   datalines;
Approve Plan    5  Drawings      Study Market  Write Specs
Drawings       10  Prototype     .             .
Study Market    5  Mkt. Strat.   .             .
Write Specs     5  Prototype     .             .
```

```
Prototype     15  Materials     Facility      .
Mkt. Strat.   10  Test Market   Marketing     .
Materials     10  Init. Prod.   .             .
Facility      10  Init. Prod.   .             .
Init. Prod.   10  Test Market   Marketing     Evaluate
Evaluate      10  Changes       .             .
Test Market   15  Changes       .             .
Changes        5  Production    .             .
Production     0  .             .             .
Marketing      0  .             .             .
;

/* Invoke PROC CPM to schedule the project specifying the */
/* ACTIVITY, DURATION and SUCCESSOR variables             */
proc cpm;
   activity task;
   duration days;
   successor succ1 succ2 succ3;
   run;

title 'Widget Manufacture: Activity-On-Node Format';
title2 'Critical Path';
proc print;
   run;
```

Output 4.1.2 Critical Path

Widget Manufacture: Activity-On-Node Format
Critical Path

Obs	task	succ1	succ2	succ3	days	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	Approve Plan	Drawings	Study Market	Write Specs	5	0	5	0	5	0	0
2	Drawings	Prototype			10	5	15	5	15	0	0
3	Study Market	Mkt. Strat.			5	5	10	35	40	30	0
4	Write Specs	Prototype			5	5	10	10	15	5	5
5	Prototype	Materials	Facility		15	15	30	15	30	0	0
6	Mkt. Strat.	Test Market	Marketing		10	10	20	40	50	30	30
7	Materials	Init. Prod.			10	30	40	30	40	0	0
8	Facility	Init. Prod.			10	30	40	30	40	0	0
9	Init. Prod.	Test Market	Marketing	Evaluate	10	40	50	40	50	0	0
10	Evaluate	Changes			10	50	60	55	65	5	5
11	Test Market	Changes			15	50	65	50	65	0	0
12	Changes	Production			5	65	70	65	70	0	0
13	Production				0	70	70	70	70	0	0
14	Marketing				0	50	50	70	70	20	20

Alternately, if you know that the project is to start on December 1, 2003, then you can determine the project schedule with reference to calendar dates by specifying the DATE= option in the PROC CPM statement. The default unit of duration is assumed to be DAY. The architecture of PROC CPM enables you to include any number of additional variables that are relevant to the project. Here, for example, you may want to include more descriptive activity names and department information. The data set DETAILS contains more information about the project that is merged with the WIDGET data set to produce the WIDGETN data set. The ID statement is useful to carry information through to the data set. Output 4.1.3 displays the resulting output data set.

```
data details;
   format task $12. dept $13. descrpt $30. ;
   input task & dept $ descrpt & ;
   label dept =  "Department"
         descrpt = "Activity Description";
   datalines;
Approve Plan  Planning       Finalize and Approve Plan
Drawings      Engineering    Prepare Drawings
Study Market  Marketing      Analyze Potential Markets
Write Specs   Engineering    Write Specifications
Prototype     Engineering    Build Prototype
Mkt. Strat.   Marketing      Develop Marketing Concept
Materials     Manufacturing  Procure Raw Materials
Facility      Manufacturing  Prepare Manufacturing Facility
Init. Prod.   Manufacturing  Initial Production Run
Evaluate      Testing        Evaluate Product In-House
Test Market   Testing        Mail Product to Sample Market
Changes       Engineering    Engineering Changes
Production    Manufacturing  Begin Full Scale Production
Marketing     Marketing      Begin Full Scale Marketing
   ;

/* Combine project network data with additional details */
data widgetn;
   merge widget details;
   run;


/* Schedule using PROC CPM, identifying the variables */
/* that specify additional project information        */
/* and set project start date to be December 1, 2003  */
proc cpm data=widgetn date='1dec03'd;
   activity task;
   successor succ1 succ2 succ3;
   duration days;
   id dept descrpt;
   run;
```

```
proc sort;
   by e_start;
   run;

title2 'Project Schedule';
proc print;
   id descrpt;
   var dept e_: l_: t_float f_float;
   run;
```

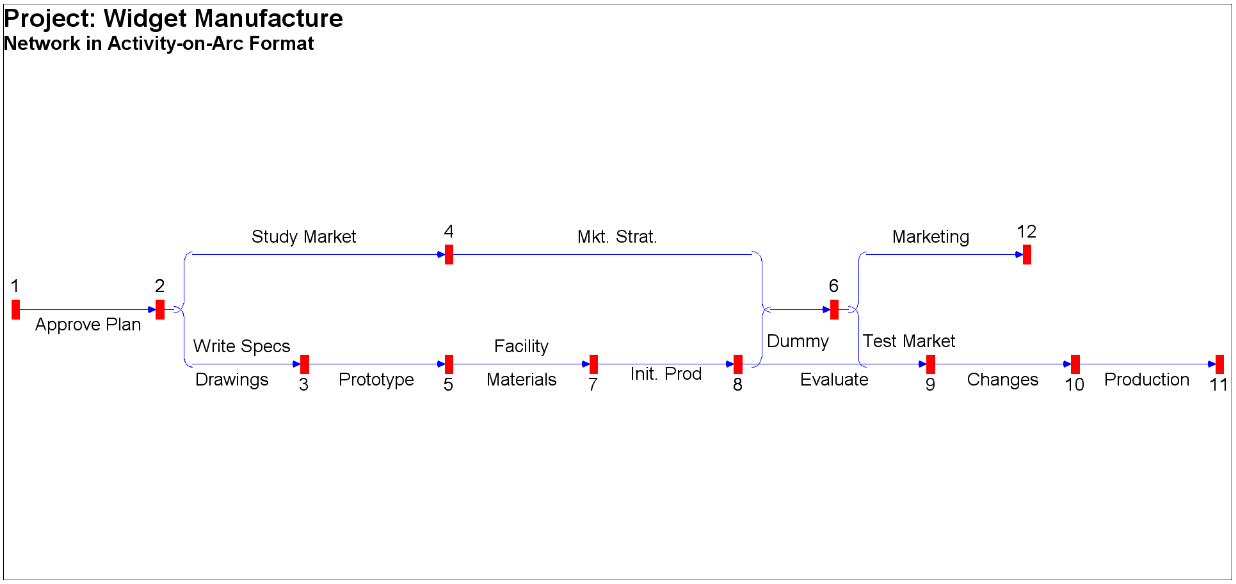
Output 4.1.3 Critical Path: Activity-On-Node Format

Widget Manufacture: Activity-On-Node Format
Project Schedule

descrpt	dept	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Finalize and Approve Plan	Planning	01DEC03	05DEC03	01DEC03	05DEC03	0	0
Prepare Drawings	Engineering	06DEC03	15DEC03	06DEC03	15DEC03	0	0
Analyze Potential Markets	Marketing	06DEC03	10DEC03	05JAN04	09JAN04	30	0
Write Specifications	Engineering	06DEC03	10DEC03	11DEC03	15DEC03	5	5
Develop Marketing Concept	Marketing	11DEC03	20DEC03	10JAN04	19JAN04	30	30
Build Prototype	Engineering	16DEC03	30DEC03	16DEC03	30DEC03	0	0
Procure Raw Materials	Manufacturing	31DEC03	09JAN04	31DEC03	09JAN04	0	0
Prepare Manufacturing Facility	Manufacturing	31DEC03	09JAN04	31DEC03	09JAN04	0	0
Initial Production Run	Manufacturing	10JAN04	19JAN04	10JAN04	19JAN04	0	0
Evaluate Product In-House	Testing	20JAN04	29JAN04	25JAN04	03FEB04	5	5
Mail Product to Sample Market	Testing	20JAN04	03FEB04	20JAN04	03FEB04	0	0
Begin Full Scale Marketing	Marketing	20JAN04	20JAN04	09FEB04	09FEB04	20	20
Engineering Changes	Engineering	04FEB04	08FEB04	04FEB04	08FEB04	0	0
Begin Full Scale Production	Manufacturing	09FEB04	09FEB04	09FEB04	09FEB04	0	0

Example 4.2: Activity-on-Arc Representation

Output 4.2.1 Network Showing Task Relationships in Activity-on-Arc Format



The problem discussed in Example 4.1 can also be described in an AOA format. The network is illustrated in Output 4.2.1. The network has an arc labeled ‘Dummy’, which is required to accurately capture all the precedence relationships. Dummy arcs are often needed when representing scheduling problems in AOA format.

The following DATA step saves the network description in a SAS data set, WIDGAOA. The data set contains the minimum amount of information required by PROC CPM for an activity network in AOA format, namely, the TAILNODE and HEADNODE variables, which indicate the direction of each arc in the network and the DURATION variable which gives the length of each task. In addition, the data set also contains a variable identifying the name of the task associated with each arc. This variable, task, can be identified to PROC CPM using the ACTIVITY statement. PROC CPM treats each observation in the data set as a new task, thus enabling you to specify multiple arcs between a pair of nodes. In this example, for instance, both the tasks ‘Drawings’ and ‘Write Specs’ connect the nodes 2 and 3; likewise, both the tasks ‘Materials’ and ‘Facility’ connect the nodes 5 and 7. If multiple arcs are not allowed, you would need more dummy arcs in this example. However, the dummy arc between nodes 8 and 6 is essential to the structure of the network and cannot be eliminated.

As in Example 4.1, the data set DETAILS containing additional activity information, can be merged with the Activity data set and used as input to PROC CPM to determine the project schedule. For purposes of display (in Gantt charts, and so on) the dummy activity has been given a label, ‘Production Milestone’. Output 4.2.2 displays the project schedule.

```
/* Activity-on-Arc representation of the project */
data widgaoa;
   format task $12. ;
   input task & days tail head;
   datalines;
```

```
Approve Plan   5   1   2
Drawings      10   2   3
Study Market   5   2   4
Write Specs    5   2   3
Prototype     15   3   5
Mkt. Strat.   10   4   6
Materials     10   5   7
Facility      10   5   7
Init. Prod.   10   7   8
Evaluate      10   8   9
Test Market   15   6   9
Changes        5   9  10
Production     0  10  11
Marketing      0   6  12
Dummy          0   8   6
;

data details;
   format task $12. dept $13. descrpt $30.;
   input task & dept $ descrpt & ;
   label dept = "Department"
         descrpt = "Activity Description";
   datalines;
Approve Plan  Planning       Finalize and Approve Plan
Drawings      Engineering    Prepare Drawings
Study Market  Marketing      Analyze Potential Markets
Write Specs   Engineering    Write Specifications
Prototype     Engineering    Build Prototype
Mkt. Strat.   Marketing      Develop Marketing Concept
Materials     Manufacturing  Procure Raw Materials
Facility      Manufacturing  Prepare Manufacturing Facility
Init. Prod.   Manufacturing  Initial Production Run
Evaluate      Testing        Evaluate Product In-House
Test Market   Testing        Mail Product to Sample Market
Changes       Engineering    Engineering Changes
Production    Manufacturing  Begin Full Scale Production
Marketing     Marketing      Begin Full Scale Marketing
Dummy         .              Production Milestone
;

data widgeta;
   merge widgaoa details;
   run;


/* The project is scheduled using PROC CPM */
/* The network information is conveyed using the TAILNODE */
/* and HEADNODE statements. The ID statement is used to   */
/* transfer project information to the output data set    */
proc cpm data=widgeta date='1dec03'd out=save;
   tailnode tail;
   headnode head;
   duration days;
   activity task;
```

```
   id dept descrpt;
   run;

proc sort;
   by e_start;
   run;

title 'Widget Manufacture: Activity-On-Arc Format';
title2 'Project Schedule';
proc print;
   id descrpt;
   var dept e_: l_: t_float f_float;
   run;
```

Output 4.2.2 Critical Path: Activity-on-Arc Format

Widget Manufacture: Activity-On-Arc Format
Project Schedule

descrpt	dept	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Finalize and Approve Plan	Planning	01DEC03	05DEC03	01DEC03	05DEC03	0	0
Prepare Drawings	Engineering	06DEC03	15DEC03	06DEC03	15DEC03	0	0
Analyze Potential Markets	Marketing	06DEC03	10DEC03	05JAN04	09JAN04	30	0
Write Specifications	Engineering	06DEC03	10DEC03	11DEC03	15DEC03	5	5
Develop Marketing Concept	Marketing	11DEC03	20DEC03	10JAN04	19JAN04	30	30
Build Prototype	Engineering	16DEC03	30DEC03	16DEC03	30DEC03	0	0
Procure Raw Materials	Manufacturing	31DEC03	09JAN04	31DEC03	09JAN04	0	0
Prepare Manufacturing Facility	Manufacturing	31DEC03	09JAN04	31DEC03	09JAN04	0	0
Initial Production Run	Manufacturing	10JAN04	19JAN04	10JAN04	19JAN04	0	0
Evaluate Product In-House	Testing	20JAN04	29JAN04	25JAN04	03FEB04	5	5
Mail Product to Sample Market	Testing	20JAN04	03FEB04	20JAN04	03FEB04	0	0
Begin Full Scale Marketing	Marketing	20JAN04	20JAN04	09FEB04	09FEB04	20	20
Production Milestone		20JAN04	20JAN04	20JAN04	20JAN04	0	0
Engineering Changes	Engineering	04FEB04	08FEB04	04FEB04	08FEB04	0	0
Begin Full Scale Production	Manufacturing	09FEB04	09FEB04	09FEB04	09FEB04	0	0

Example 4.3: Meeting Project Deadlines

This example illustrates the use of the project finish date (using the FBDATE= option) to specify a deadline on the project. In the following program it is assumed that the project data are saved in the data set WIDGAOA. PROC CPM is first invoked with the FBDATE= option. Output 4.3.1 shows the resulting schedule. The entire schedule is shifted in time (as compared to the schedule in Output 4.2.2) so that the end of the project is on March 1, 2004. The second part of the program specifies a project start date in addition to the project finish date using both the DATE= and FBDATE= options. The schedule displayed in Output 4.3.2 shows that all of the activities have a larger float than before due to the imposition of a less stringent target date.

```
proc cpm data=widgaoa
         fbdate='1mar04'd interval=day;
   tailnode tail;
   headnode head;
   duration days;
   id task;
   run;

proc sort;
   by e_start;
   run;

title 'Meeting Project Deadlines';
title2 'Specification of Project Finish Date';
proc print;
   id task;
   var e_: l_: t_float f_float;
   run;


proc cpm data=widgaoa
         fbdate='1mar04'd
         date='1dec03'd interval=day;
   tailnode tail;
   headnode head;
   duration days;
   id task;
   run;

proc sort;
   by e_start;
   run;

title2 'Specifying Project Start and Completion Dates';
proc print;
   id task;
   var e_: l_: t_float f_float;
   run;
```

Output 4.3.1 Meeting Project Deadlines: FBDATE= Option

Meeting Project Deadlines
Specification of Project Finish Date

task	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Approve Plan	22DEC03	26DEC03	22DEC03	26DEC03	0	0
Drawings	27DEC03	05JAN04	27DEC03	05JAN04	0	0
Study Market	27DEC03	31DEC03	26JAN04	30JAN04	30	0
Write Specs	27DEC03	31DEC03	01JAN04	05JAN04	5	5
Mkt. Strat.	01JAN04	10JAN04	31JAN04	09FEB04	30	30
Prototype	06JAN04	20JAN04	06JAN04	20JAN04	0	0
Materials	21JAN04	30JAN04	21JAN04	30JAN04	0	0
Facility	21JAN04	30JAN04	21JAN04	30JAN04	0	0
Init. Prod.	31JAN04	09FEB04	31JAN04	09FEB04	0	0
Evaluate	10FEB04	19FEB04	15FEB04	24FEB04	5	5
Test Market	10FEB04	24FEB04	10FEB04	24FEB04	0	0
Marketing	10FEB04	10FEB04	01MAR04	01MAR04	20	20
Dummy	10FEB04	10FEB04	10FEB04	10FEB04	0	0
Changes	25FEB04	29FEB04	25FEB04	29FEB04	0	0
Production	01MAR04	01MAR04	01MAR04	01MAR04	0	0

Output 4.3.2 Meeting Project Deadlines: DATE= and FBDATE= Options

Meeting Project Deadlines
Specifying Project Start and Completion Dates

task	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Approve Plan	01DEC03	05DEC03	22DEC03	26DEC03	21	0
Drawings	06DEC03	15DEC03	27DEC03	05JAN04	21	0
Study Market	06DEC03	10DEC03	26JAN04	30JAN04	51	0
Write Specs	06DEC03	10DEC03	01JAN04	05JAN04	26	5
Mkt. Strat.	11DEC03	20DEC03	31JAN04	09FEB04	51	30
Prototype	16DEC03	30DEC03	06JAN04	20JAN04	21	0
Materials	31DEC03	09JAN04	21JAN04	30JAN04	21	0
Facility	31DEC03	09JAN04	21JAN04	30JAN04	21	0
Init. Prod.	10JAN04	19JAN04	31JAN04	09FEB04	21	0
Evaluate	20JAN04	29JAN04	15FEB04	24FEB04	26	5
Test Market	20JAN04	03FEB04	10FEB04	24FEB04	21	0
Marketing	20JAN04	20JAN04	01MAR04	01MAR04	41	41
Dummy	20JAN04	20JAN04	10FEB04	10FEB04	21	0
Changes	04FEB04	08FEB04	25FEB04	29FEB04	21	0
Production	09FEB04	09FEB04	01MAR04	01MAR04	21	21

Example 4.4: Displaying the Schedule on a Calendar

This example shows how you can use the output from CPM to display calendars containing the critical path schedule and the early start schedule. The example uses the network described in Example 4.2 and assumes that the data set SAVE contains the project schedule. The following program invokes PROC CALENDAR to produce two calendars; the first calendar in Output 4.4.1 displays only the critical activities in the project, while the second calendar in Output 4.4.1 displays all the activities in the project. In both invocations of PROC CALENDAR, a WHERE statement is used to display only the activities that are scheduled to finish in December.

```
proc cpm data=widgaoa out=save
   date='1dec03'd interval=day;
   tailnode tail;
   headnode head;
   duration days;
   id task;
   run;

proc sort data=save out=crit;
   where t_float=0;
   by e_start;
   run;

title 'Printing the Schedule on a Calendar';
title2 'Critical Activities in December';

/* print the critical act. calendar  */
options nodate pageno=1 pagesize=50;

proc calendar schedule
     data=crit;
   id e_start;
   where e_finish <= '31dec03'd;
   var task;
   dur days;
   run;

/* sort data for early start calendar */
proc sort data=save;
   by e_start;

/* print the early start calendar */
title2 'Early Start Schedule for December';
options nodate pageno=1 pagesize=50;
proc calendar schedule data=save;
   id e_start;
   where e_finish <= '31dec03'd;
   var task;
   dur days;
   run;
```

Output 4.4.1 Project Calendar: All Activities

```
                    Printing the Schedule on a Calendar
                      Critical Activities in December

-------------------------------------------------------------------------------
|                                                                             |
|                              December  2003                                 |
|                                                                             |
|-----------------------------------------------------------------------------|
|  Sunday  |  Monday  | Tuesday  |Wednesday | Thursday |  Friday  | Saturday |
|----------+----------+----------+----------+----------+----------+----------|
|          |    1     |    2     |    3     |    4     |    5     |    6     |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|          |+===================Approve Plan===================+|+Drawings>|
|----------+----------+----------+----------+----------+----------+----------|
|    7     |    8     |    9     |    10    |    11    |    12    |    13    |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|<===============================Drawings===============================>|
|----------+----------+----------+----------+----------+----------+----------|
|    14    |    15    |    16    |    17    |    18    |    19    |    20    |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|<=====Drawings======+|+===================Prototype====================>|
|----------+----------+----------+----------+----------+----------+----------|
|    21    |    22    |    23    |    24    |    25    |    26    |    27    |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|<==============================Prototype===============================>|
|----------+----------+----------+----------+----------+----------+----------|
|    28    |    29    |    30    |    31    |          |          |          |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|<==========Prototype==========+|          |          |          |          |
-------------------------------------------------------------------------------
```

Output 4.4.1 *continued*

Printing the Schedule on a Calendar
Early Start Schedule for December

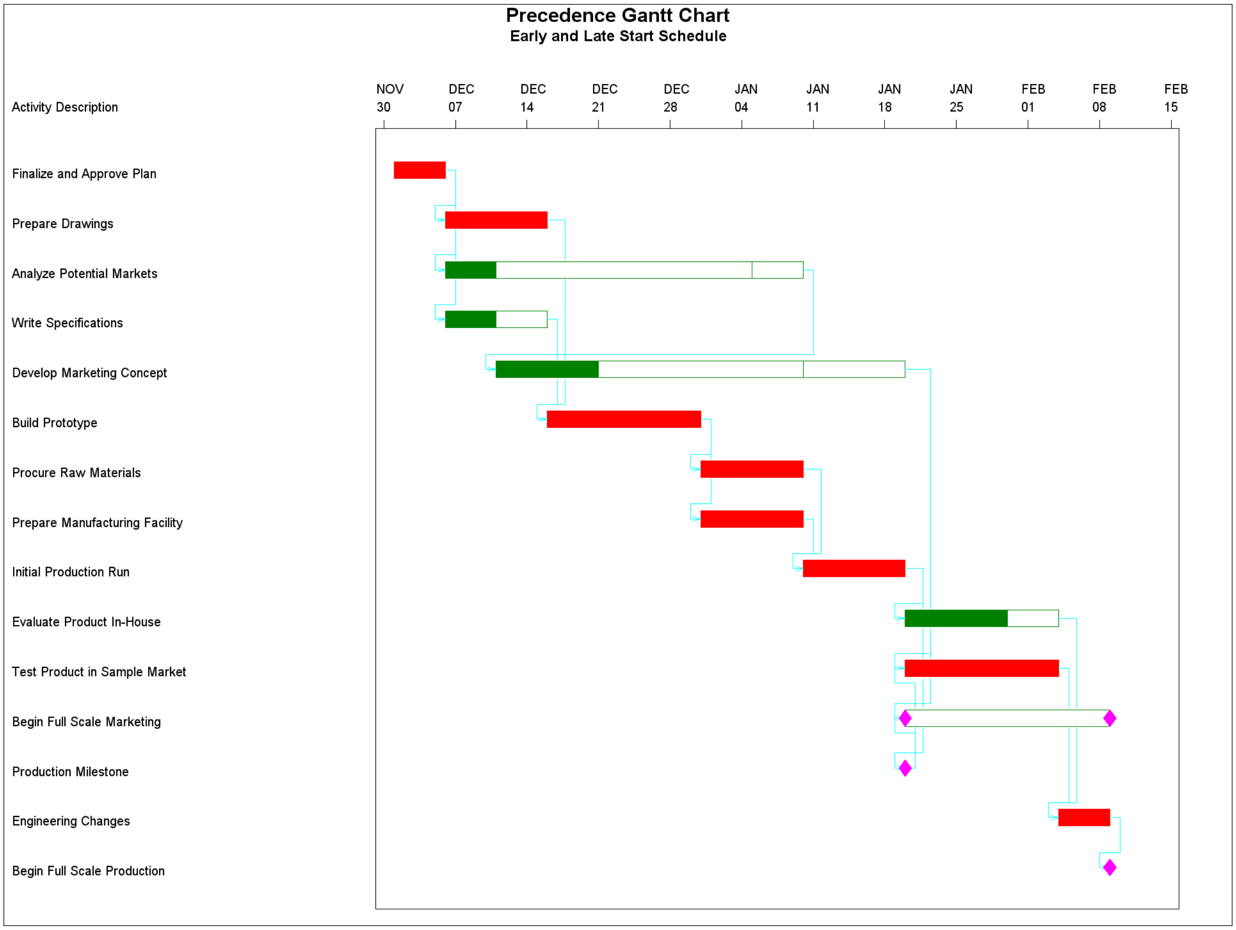
```
------------------------------------------------------------------------------
|                                                                            |
|                               December  2003                               |
|                                                                            |
|----------------------------------------------------------------------------|
|  Sunday  |  Monday  | Tuesday  |Wednesday | Thursday |  Friday  | Saturday |
|----------+----------+----------+----------+----------+----------+----------|
|          |    1     |    2     |    3     |    4     |    5     |    6     |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |+Write Sp>|
|          |          |          |          |          |          |+Study Ma>|
|          |+====================Approve Plan====================+|+Drawings>|
|----------+----------+----------+----------+----------+----------+----------|
|    7     |    8     |    9     |    10    |    11    |    12    |    13    |
|          |          |          |          |          |          |          |
|<===============Write Specs===============+|          |          |          |
|<==============Study Market===============+|+=========Mkt. Strat.==========>|
|<=================================Drawings=================================>|
|----------+----------+----------+----------+----------+----------+----------|
|    14    |    15    |    16    |    17    |    18    |    19    |    20    |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|<===============================Mkt. Strat.================================+|
|<=====Drawings======+|+=====================Prototype======================>|
|----------+----------+----------+----------+----------+----------+----------|
|    21    |    22    |    23    |    24    |    25    |    26    |    27    |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|<================================Prototype=================================>|
|----------+----------+----------+----------+----------+----------+----------|
|    28    |    29    |    30    |    31    |          |          |          |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|          |          |          |          |          |          |          |
|<==========Prototype===========+|          |          |          |          |
------------------------------------------------------------------------------
```

Example 4.5: Precedence Gantt Chart

This example produces a Gantt chart of the schedule obtained from PROC CPM. The example uses the network described in Example 4.2 (AOA format) and assumes that the data set SAVE contains the schedule produced by PROC CPM and sorted by the variable E_START. The Gantt chart produced shows the early and late start schedules as well as the precedence relationships between the activities. The precedence information is conveyed to PROC GANTT through the TAILNODE= and HEADNODE= options.

```
data details;
   input task $ 1-12 dept $ 15-27 descrpt $ 30-59;
   label dept = "Department"
         descrpt = "Activity Description";
   datalines;
Dev. Concept  Planning       Finalize and Approve Plan
Drawings      Engineering    Prepare Drawings
Study Market  Marketing      Analyze Potential Markets
Write Specs   Engineering    Write Specifications
Prototype     Engineering    Build Prototype
Mkt. Strat.   Marketing      Develop Marketing Concept
Materials     Manufacturing  Procure Raw Materials
Facility      Manufacturing  Prepare Manufacturing Facility
Init. Prod.   Manufacturing  Initial Production Run
Evaluate      Testing        Evaluate Product In-House
Test Market   Testing        Test Product in Sample Market
Changes       Engineering    Engineering Changes
Production    Manufacturing  Begin Full Scale Production
Marketing     Marketing      Begin Full Scale Marketing
Dummy                        Production Milestone
;

data widgeta;
   merge widgaoa details;
   run;


* specify the device on which you want the chart printed;

goptions vpos=50 hpos=80 border;

title 'Precedence Gantt Chart';
title2 'Early and Late Start Schedule';

proc gantt graphics data=save;
   chart / compress tailnode=tail headnode=head
           height=2 nojobnum skip=2
           cprec=cyan cmile=magenta
           caxis=black
           dur=days increment=7 nolegend;
   id descrpt;
   run;
```

Output 4.5.1 Gantt Chart of Project



Example 4.6: Changing Duration Units

This example illustrates the use of the INTERVAL= option to identify the units of duration to PROC CPM. In the previous examples, it was assumed that work can be done on the activities all seven days of the week without any break. Suppose now that you want to schedule the activities only on weekdays. To do so, specify INTERVAL=WEEKDAY in the PROC CPM statement. Output 4.6.1 displays the schedule produced by PROC CPM. Note that, with a shorter work week, the project finishes on March 8, 2004, instead of on February 9, 2004.

```
proc cpm data=widget out=save
     date='1dec03'd interval=weekday;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   run;

title 'Changing Duration Units';
title2 'INTERVAL=WEEKDAY';
proc print;
   id task;
   var e_: l_: t_float f_float;
   run;
```

Output 4.6.1 Changing Duration Units: INTERVAL=WEEKDAY

Changing Duration Units
INTERVAL=WEEKDAY

task	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Approve Plan	01DEC03	05DEC03	01DEC03	05DEC03	0	0
Drawings	08DEC03	19DEC03	08DEC03	19DEC03	0	0
Study Market	08DEC03	12DEC03	19JAN04	23JAN04	30	0
Write Specs	08DEC03	12DEC03	15DEC03	19DEC03	5	5
Prototype	22DEC03	09JAN04	22DEC03	09JAN04	0	0
Mkt. Strat.	15DEC03	26DEC03	26JAN04	06FEB04	30	30
Materials	12JAN04	23JAN04	12JAN04	23JAN04	0	0
Facility	12JAN04	23JAN04	12JAN04	23JAN04	0	0
Init. Prod.	26JAN04	06FEB04	26JAN04	06FEB04	0	0
Evaluate	09FEB04	20FEB04	16FEB04	27FEB04	5	5
Test Market	09FEB04	27FEB04	09FEB04	27FEB04	0	0
Changes	01MAR04	05MAR04	01MAR04	05MAR04	0	0
Production	08MAR04	08MAR04	08MAR04	08MAR04	0	0
Marketing	09FEB04	09FEB04	08MAR04	08MAR04	20	20

To display the weekday schedule on a calendar, use the WEEKDAY option in the PROC CALENDAR statement. The following code sorts the Schedule data set by the E_START variable and produces a calendar shown in Output 4.6.2, which displays the schedule of activities for the month of December.

```
proc sort;
   by e_start;
   run;

/* truncate schedule: print only for december */
data december;
   set save;
   e_finish = min('31dec03'd, e_finish);
   if e_start <= '31dec03'd;
   run;

title3 'Calendar of Schedule';
options nodate pageno=1 ps=50;
proc calendar data=december schedule weekdays;
   id e_start;
   finish e_finish;
   var task;
   run;
```

Output 4.6.2 Changing Duration Units: WEEKDAY Calendar for December

Changing Duration Units
INTERVAL=WEEKDAY
Calendar of Schedule

```
--------------------------------------------------------------------------------
|                                                                              |
|                              December  2003                                  |
|                                                                              |
|------------------------------------------------------------------------------|
|    Monday      |    Tuesday     |   Wednesday    |   Thursday     |    Friday      |
|----------------+----------------+----------------+----------------+----------------|
|       1        |       2        |       3        |       4        |       5        |
|                |                |                |                |                |
|                |                |                |                |                |
|                |                |                |                |                |
|                |                |                |                |                |
|+==============================Approve Plan===============================+|
|----------------+----------------+----------------+----------------+----------------|
|       8        |       9        |      10        |      11        |      12        |
|                |                |                |                |                |
|                |                |                |                |                |
|+===============================Write Specs===============================+|
|+==============================Study Market===============================+|
|+================================Drawings=================================>|
|----------------+----------------+----------------+----------------+----------------|
|      15        |      16        |      17        |      18        |      19        |
|                |                |                |                |                |
|                |                |                |                |                |
|                |                |                |                |                |
|+===============================Mkt. Strat.===============================>|
|<================================Drawings=================================+|
|----------------+----------------+----------------+----------------+----------------|
|      22        |      23        |      24        |      25        |      26        |
|                |                |                |                |                |
|                |                |                |                |                |
|                |                |                |                |                |
|+================================Prototype================================>|
|<===============================Mkt. Strat.===============================+|
|----------------+----------------+----------------+----------------+----------------|
|      29        |      30        |      31        |                |                |
|                |                |                |                |                |
|                |                |                |                |                |
|                |                |                |                |                |
|                |                |                |                |                |
|<=================Prototype=================+|                |                |
--------------------------------------------------------------------------------
```

The durations of the activities in the project are multiples of 5. Thus, if work is done only on weekdays, all activities in the project last 0, 1, 2, or 3 weeks. The INTERVAL= option can also be used to set the units of duration to hours, minutes, seconds, years, months, quarters, or weeks. In this example, the data set WIDGWK is created from WIDGET to set the durations in weeks. PROC CPM is then invoked with INTERVAL=WEEK, and the resulting schedule is displayed in Output 4.6.3. Note that the float values are also expressed in units of weeks.

```
data widgwk;
   set widget;
   weeks = days / 5;
   run;

proc cpm data=widgwk date='1dec03'd interval=week;
   activity  task;
   successor succ1 succ2 succ3;
   duration  weeks;
   id task;
   run;

title2 'INTERVAL=WEEK';
proc print;
   id task;
   var e_: l_: t_float f_float;
   run;
```

Output 4.6.3 Changing Duration Units: INTERVAL=WEEK

Changing Duration Units
INTERVAL=WEEK

task	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Approve Plan	01DEC03	07DEC03	01DEC03	07DEC03	0	0
Drawings	08DEC03	21DEC03	08DEC03	21DEC03	0	0
Study Market	08DEC03	14DEC03	19JAN04	25JAN04	6	0
Write Specs	08DEC03	14DEC03	15DEC03	21DEC03	1	1
Prototype	22DEC03	11JAN04	22DEC03	11JAN04	0	0
Mkt. Strat.	15DEC03	28DEC03	26JAN04	08FEB04	6	6
Materials	12JAN04	25JAN04	12JAN04	25JAN04	0	0
Facility	12JAN04	25JAN04	12JAN04	25JAN04	0	0
Init. Prod.	26JAN04	08FEB04	26JAN04	08FEB04	0	0
Evaluate	09FEB04	22FEB04	16FEB04	29FEB04	1	1
Test Market	09FEB04	29FEB04	09FEB04	29FEB04	0	0
Changes	01MAR04	07MAR04	01MAR04	07MAR04	0	0
Production	08MAR04	08MAR04	08MAR04	08MAR04	0	0
Marketing	09FEB04	09FEB04	08MAR04	08MAR04	4	4

Example 4.7: Controlling the Project Calendar

This example illustrates the use of the INTERVAL=, DAYSTART=, and DAYLENGTH= options to control the project calendar. In Example 4.1 through Example 4.5, none of these three options is specified; hence the durations are assumed to be days (INTERVAL=DAY), and work is scheduled on all seven days of the week. In Example 4.6, the specification of INTERVAL=WEEKDAY causes the schedule to skip weekends. The present example shows further ways of controlling the project calendar. For example, you may want to control the work pattern during a standard week or the start and length of the workday.

Suppose you want to schedule the project specified in Example 4.1 but you want to schedule only on weekdays from 9 a.m. to 5 p.m. To schedule the project, use the INTERVAL=WORKDAY option rather than the default INTERVAL=DAY. Then, one unit of duration is interpreted as eight hours of work. To schedule the manufacturing project to start on December 1, with an eight-hour workday and a five-day work week, you can invoke PROC CPM with the following statements. Output 4.7.1 displays the resulting schedule; the start and finish times are expressed in SAS datetime values.

```
title 'Controlling the Project Calendar';
title2 'Scheduling on Workdays';
proc cpm data=widget date='1dec03'd interval=workday;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   run;

title3 'Day Starts at 9 a.m.';
proc print;
   id task;
   var e_: l_: t_float f_float;
   run;
```

Output 4.7.1 Controlling the Project Calendar: INTERVAL=WORKDAY

Controlling the Project Calendar
Scheduling on Workdays
Day Starts at 9 a.m.

task	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Approve Plan	01DEC03:09:00:00	05DEC03:16:59:59	01DEC03:09:00:00	05DEC03:16:59:59	0	0
Drawings	08DEC03:09:00:00	19DEC03:16:59:59	08DEC03:09:00:00	19DEC03:16:59:59	0	0
Study Market	08DEC03:09:00:00	12DEC03:16:59:59	19JAN04:09:00:00	23JAN04:16:59:59	30	0
Write Specs	08DEC03:09:00:00	12DEC03:16:59:59	15DEC03:09:00:00	19DEC03:16:59:59	5	5
Prototype	22DEC03:09:00:00	09JAN04:16:59:59	22DEC03:09:00:00	09JAN04:16:59:59	0	0
Mkt. Strat.	15DEC03:09:00:00	26DEC03:16:59:59	26JAN04:09:00:00	06FEB04:16:59:59	30	30
Materials	12JAN04:09:00:00	23JAN04:16:59:59	12JAN04:09:00:00	23JAN04:16:59:59	0	0
Facility	12JAN04:09:00:00	23JAN04:16:59:59	12JAN04:09:00:00	23JAN04:16:59:59	0	0
Init. Prod.	26JAN04:09:00:00	06FEB04:16:59:59	26JAN04:09:00:00	06FEB04:16:59:59	0	0
Evaluate	09FEB04:09:00:00	20FEB04:16:59:59	16FEB04:09:00:00	27FEB04:16:59:59	5	5
Test Market	09FEB04:09:00:00	27FEB04:16:59:59	09FEB04:09:00:00	27FEB04:16:59:59	0	0
Changes	01MAR04:09:00:00	05MAR04:16:59:59	01MAR04:09:00:00	05MAR04:16:59:59	0	0
Production	08MAR04:09:00:00	08MAR04:09:00:00	08MAR04:09:00:00	08MAR04:09:00:00	0	0
Marketing	09FEB04:09:00:00	09FEB04:09:00:00	08MAR04:09:00:00	08MAR04:09:00:00	20	20

If you want to change the length of the workday, use the DAYLENGTH= option in the PROC CPM statement. For example, if you want an eight-and-a-half hour workday instead of the default eight-hour workday, you should include DAYLENGTH='08:30'T in the PROC CPM statement. In addition, you might also want to change the start of the workday. The workday starts at 9 a.m., by default. To change the default, use the DAYSTART= option. The following program schedules the project to start at 7 a.m. on December 1. The project is scheduled on eight-and-a-half hour workdays each starting at 7 a.m. Output 4.7.2 displays the resulting schedule produced by PROC CPM.

```
proc cpm data=widget date='1dec03'd interval=workday
         daylength='08:30't daystart='07:00't;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   run;

title3 'Day Starts at 7 a.m. and is 8.5 Hours Long';
proc print;
   id task;
   var e_: l_: t_float f_float;
   run;
```

Output 4.7.2 Controlling the Project Calendar: DAYSTART and DAYLENGTH

Controlling the Project Calendar
Scheduling on Workdays
Day Starts at 7 a.m. and is 8.5 Hours Long

task	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Approve Plan	01DEC03:07:00:00	05DEC03:15:29:59	01DEC03:07:00:00	05DEC03:15:29:59	0	0
Drawings	08DEC03:07:00:00	19DEC03:15:29:59	08DEC03:07:00:00	19DEC03:15:29:59	0	0
Study Market	08DEC03:07:00:00	12DEC03:15:29:59	19JAN04:07:00:00	23JAN04:15:29:59	30	0
Write Specs	08DEC03:07:00:00	12DEC03:15:29:59	15DEC03:07:00:00	19DEC03:15:29:59	5	5
Prototype	22DEC03:07:00:00	09JAN04:15:29:59	22DEC03:07:00:00	09JAN04:15:29:59	0	0
Mkt. Strat.	15DEC03:07:00:00	26DEC03:15:29:59	26JAN04:07:00:00	06FEB04:15:29:59	30	30
Materials	12JAN04:07:00:00	23JAN04:15:29:59	12JAN04:07:00:00	23JAN04:15:29:59	0	0
Facility	12JAN04:07:00:00	23JAN04:15:29:59	12JAN04:07:00:00	23JAN04:15:29:59	0	0
Init. Prod.	26JAN04:07:00:00	06FEB04:15:29:59	26JAN04:07:00:00	06FEB04:15:29:59	0	0
Evaluate	09FEB04:07:00:00	20FEB04:15:29:59	16FEB04:07:00:00	27FEB04:15:29:59	5	5
Test Market	09FEB04:07:00:00	27FEB04:15:29:59	09FEB04:07:00:00	27FEB04:15:29:59	0	0
Changes	01MAR04:07:00:00	05MAR04:15:29:59	01MAR04:07:00:00	05MAR04:15:29:59	0	0
Production	08MAR04:07:00:00	08MAR04:07:00:00	08MAR04:07:00:00	08MAR04:07:00:00	0	0
Marketing	09FEB04:07:00:00	09FEB04:07:00:00	08MAR04:07:00:00	08MAR04:07:00:00	20	20

An alternate way of specifying the start of each working day is to set the INTERVAL= option to DTWRKDAY and specify a SAS datetime value for the project start date. Using INTERVAL=DTWRKDAY tells CPM that the DATE= option is a SAS datetime value and that the time given is the start of the workday. For the present example, you could have used DATE='1dec03:07:00'dt in conjunction with the specification INTERVAL=DTWRKDAY and DAYLENGTH='08:30't.

Example 4.8: Scheduling around Holidays

This example shows how you can schedule around holidays with PROC CPM. First, save a list of holidays in a SAS data set as SAS date variables. The length of the holidays is assumed to be measured in units specified by the INTERVAL= option. By default, all holidays are assumed to be one unit long. You can control the length of each holiday by specifying either the finish time for each holiday or the length of each holiday in the same observation as the holiday specification.

Output 4.8.1 Scheduling around Holidays: HOLIDAYS Data Set

Scheduling Around Holidays
Data Set HOLIDAYS

Obs	holiday	holifin	holidur
1	24DEC03	26DEC03	4
2	01JAN04	.	.

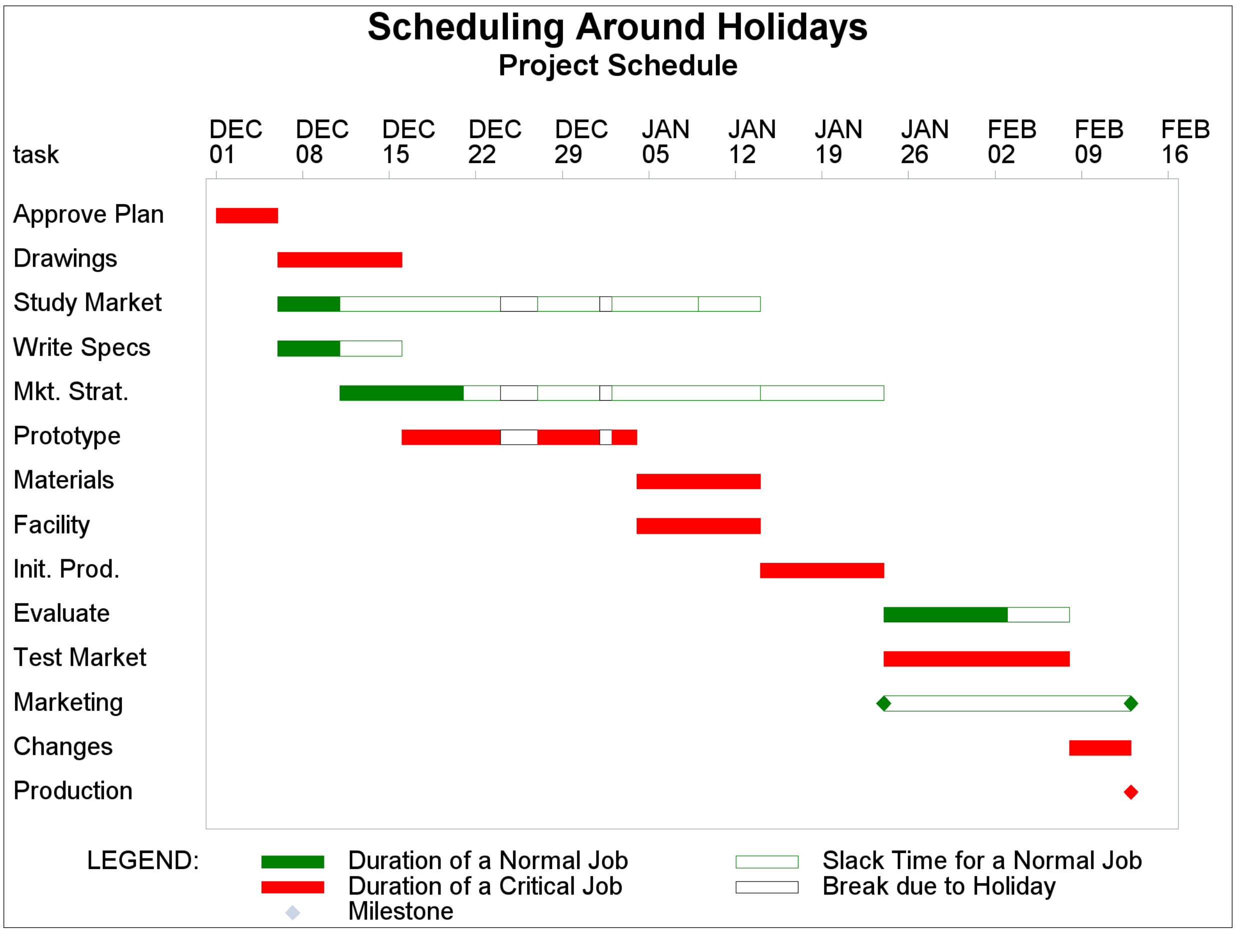
For example, the data set HOLIDAYS, displayed in Output 4.8.1 specifies two holidays, one for Christmas and the other for New Year's Day. The variable holiday specifies the start of each holiday. The variable holifin specifies the end of the Christmas holiday as 26Dec03. Alternately, the variable holidur can be used to interpret the Christmas holiday as lasting four interval units starting from the 24th of December. If the variable holidur is used, the actual days when work is not done depends on the INTERVAL= option and on the underlying calendar used. This form of specifying holidays or breaks is useful for indicating vacations for specific employees. The second observation in the data set defines the New Year's holiday as just one day long because both the variables holifin and holidur variables have missing values.

To invoke PROC CPM to schedule around holidays, use the HOLIDATA= option in the PROC CPM statement (see the following program) to identify the data set, and list the names of the variables in the data set in a HOLIDAY statement. The holiday start and finish are identified by specifying the HOLIDAY and HOLIFIN variables. Output 4.8.2 displays the schedule obtained.

```
proc cpm data=widget holidata=holidays
         out=saveh date='1dec03'd ;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   run;

proc sort data=saveh;
   by e_start;
   run;
pattern1 c=green v=s;     /* duration of a non-critical activity   */
pattern2 c=green v=e;     /* slack time for a noncrit. activity    */
pattern3 c=red   v=s;     /* duration of a critical activity       */
pattern4 c=magenta v=e;   /* slack time for a supercrit. activity  */
pattern5 c=magenta v=s;   /* duration of a supercrit. activity     */
pattern6 c=cyan v=s;      /* actual duration of an activity        */
pattern7 c=black v=e;     /* break due to a holiday                */

goptions vpos=50 hpos=80 border;
```

```
title 'Scheduling Around Holidays';
title2 'Project Schedule';

proc gantt graphics data=saveh holidata=holidays;
   chart / compress
           height=1.7 nojobnum skip=2
           dur=days increment=7
           holiday=(holiday) holifin=(holifin);

   id task;
   run;
```

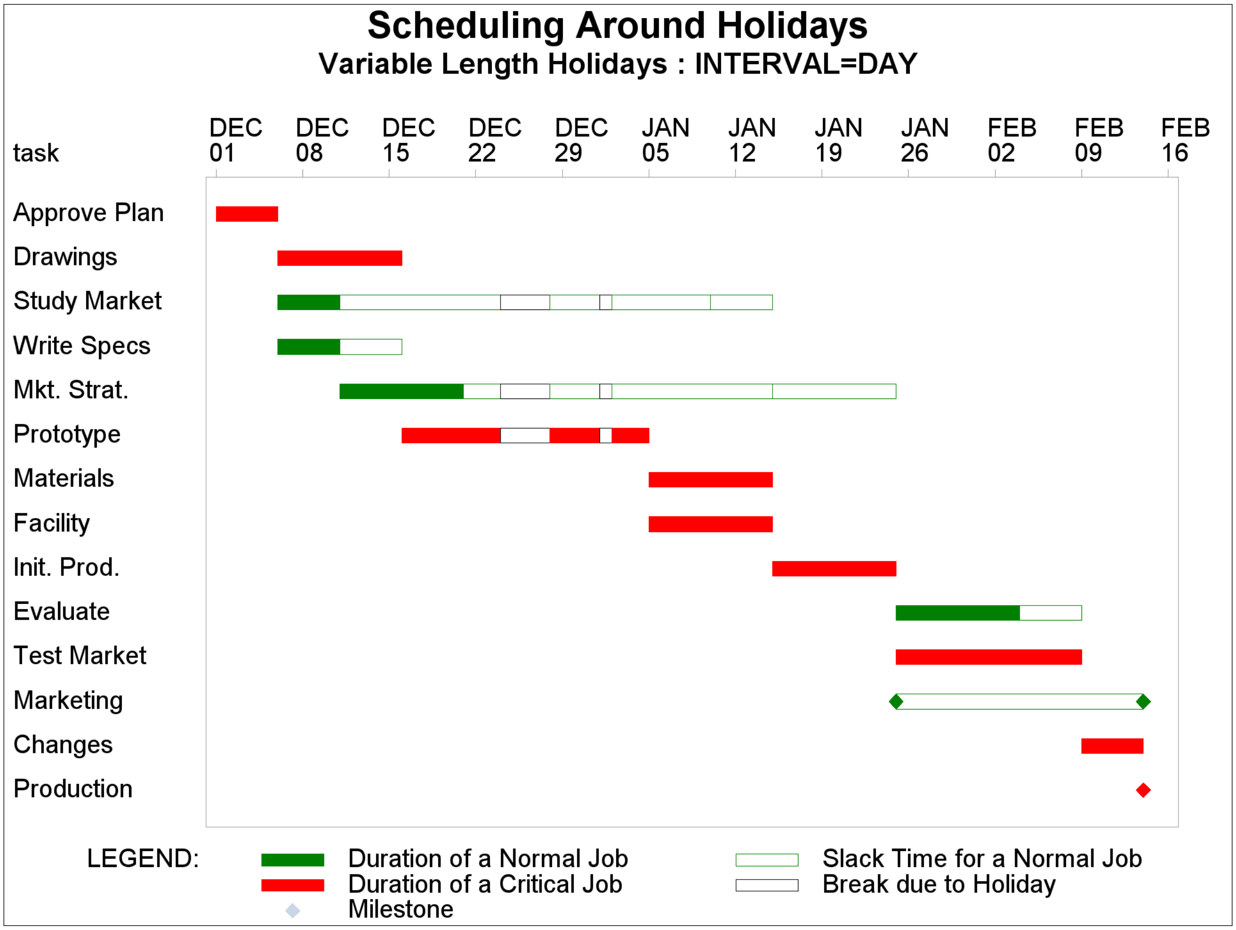
Output 4.8.2 Scheduling around Holidays: Project Schedule



The next two invocations illustrate the use of the HOLIDUR= option and the effect of the INTERVAL= option on the duration of the holidays. Recall that the holiday duration is also assumed to be in *interval* units where *interval* is the value specified for the INTERVAL= option. Suppose that a holiday period for the entire project starts on December 24, 2003, with duration specified as 4. First the project is scheduled with INTERVAL=DAY so that the holidays are on December 24, 25, 26, and 27, 2003. Output 4.8.3 displays the resulting schedule. The project completion is delayed by one day due to the extra holiday on December 27, 2003.

```
proc cpm data=widget holidata=holidays
         out=saveh1 date='1dec03'd
         interval=day;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holidur=(holidur);
   run;

title2 'Variable Length Holidays : INTERVAL=DAY';
proc sort data=saveh1;
   by e_start;
   run;

proc gantt graphics data=saveh1 holidata=holidays;
   chart / compress
           height=1.7 skip=2
           nojobnum
           dur=days increment=7
           holiday=(holiday) holidur=(holidur) interval=day;

   id task;
   run;
```

Output 4.8.3 Scheduling around Holidays: INTERVAL=DAY

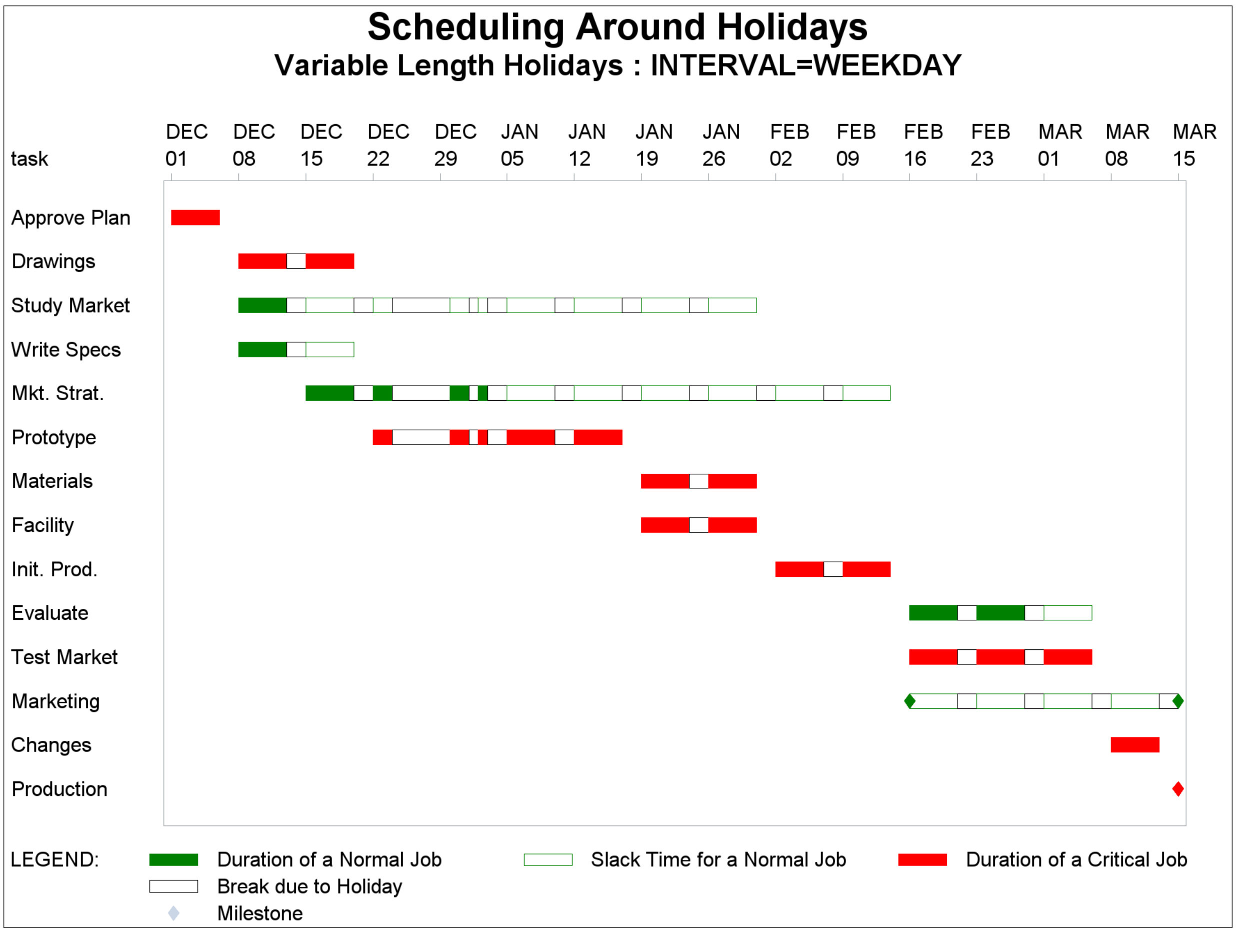


Next, suppose that work on the project is to be scheduled only on weekdays. The INTERVAL= option is set to WEEKDAY. Then, the value '4' specified for the variable holidur is interpreted as 4 weekdays. Thus, the holidays are on December 24, 25, 26, and 29, 2003, because December 27 and 28 (Saturday and Sunday) are non-working days anyway. (Note that if holifin had been used, the holiday would have ended on December 26, 2003.) The following statements schedule the project to start on December 1, 2003 with INTERVAL=WEEKDAY. Output 4.8.4 displays the resulting schedule. Note the further delay in project completion time.

```
proc cpm data=widget holidata=holidays
         out=saveh2 date='1dec03'd
         interval=weekday;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holidur=(holidur);
   run;

proc sort data=saveh2;
   by e_start;
   run;
```

```
title2 'Variable Length Holidays : INTERVAL=WEEKDAY';
proc gantt graphics data=saveh2 holidata=holidays;
   chart / compress
           height=1.8 skip=2
           nojobnum
           dur=days increment=7
           holiday=(holiday)
           holidur=(holidur)
           interval=weekday;

   id task;
   run;
```

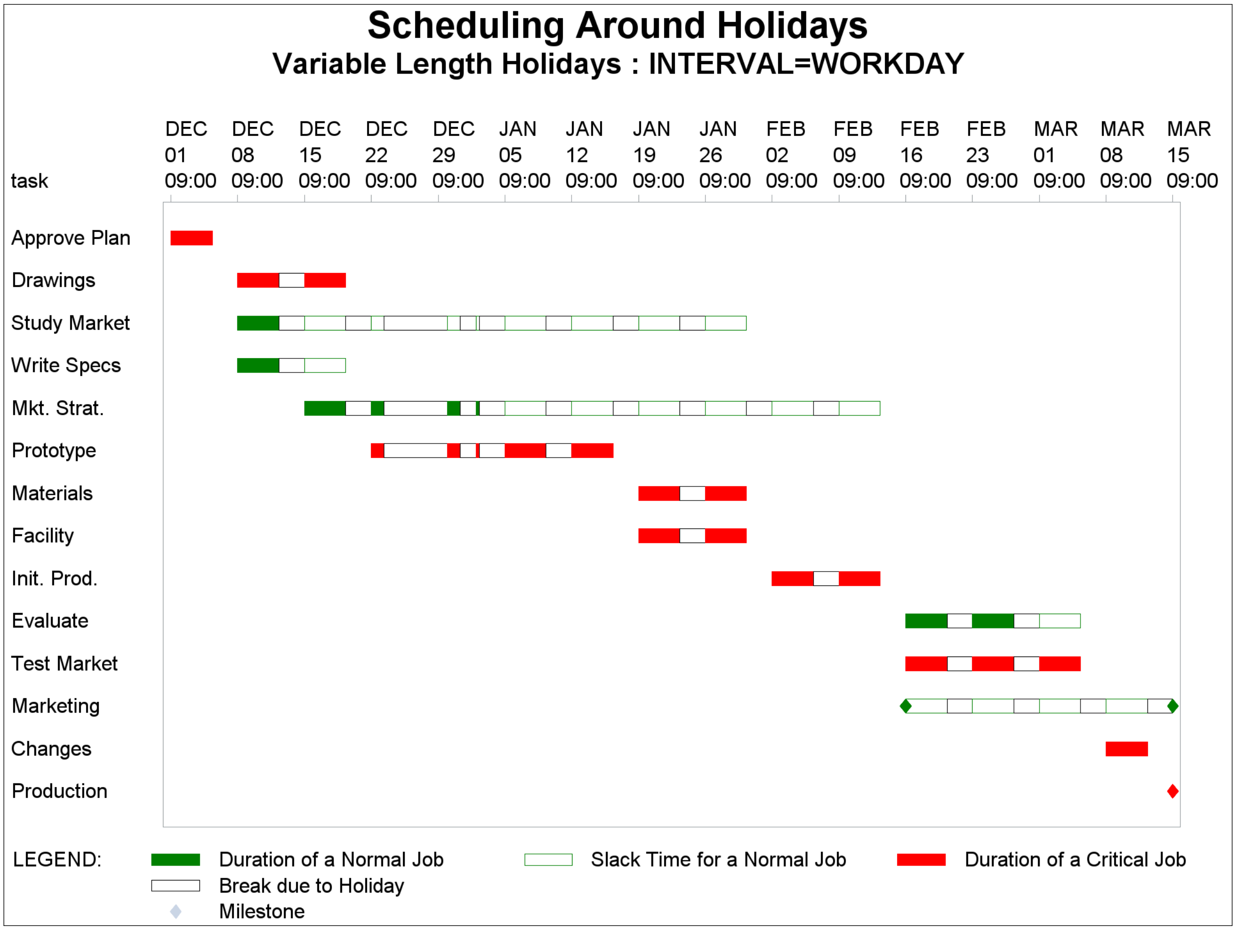
Output 4.8.4 Scheduling around Holidays: INTERVAL=WEEKDAY



Finally, the same project is scheduled to start on December 1, 2003 with INTERVAL=WORKDAY. Output 4.8.5 displays the resulting Schedule data set. This time the holiday period starts at 5:00 p.m. on December 23, 2003, and ends at 9:00 a.m. on December 30, 2003.

```
proc cpm data=widget holidata=holidays
         out=saveh3 date='1dec03'd
         interval=workday;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holidur=(holidur);
   run;

proc sort data=saveh3;
   by e_start;
   run;

title2 'Variable Length Holidays : INTERVAL=WORKDAY';
proc gantt graphics data=saveh3 holidata=holidays;
   chart / compress
           height=1.8 nojobnum skip=2
           dur=days increment=7
           holiday=(holiday) holidur=(holidur) interval=workday;
   id task;
   run;
```

Output 4.8.5 Scheduling around Holidays: INTERVAL=WORKDAY



Example 4.9: CALEDATA and WORKDATA Data Sets

This example shows how you can schedule the job over a nonstandard day and a nonstandard week. In the first part of the example, the calendar followed is a six-day week with an eight-and-a-half hour workday starting at 7 a.m. The project data are the same as were used in Example 4.8, but some of the durations have been changed to include some fractional values. Output 4.9.1 shows the project data set.

Output 4.9.1 Data Set WIDGET9: Scheduling on the Six-Day Week

Scheduling on the 6-Day Week
Data Set WIDGET9

Obs	task	days	succ1	succ2	succ3
1	Approve Plan	5.5	Drawings	Study Market	Write Specs
2	Drawings	10.0	Prototype		
3	Study Market	5.0	Mkt. Strat.		
4	Write Specs	4.5	Prototype		
5	Prototype	15.0	Materials	Facility	
6	Mkt. Strat.	10.0	Test Market	Marketing	
7	Materials	10.0	Init. Prod.		
8	Facility	10.0	Init. Prod.		
9	Init. Prod.	10.0	Test Market	Marketing	Evaluate
10	Evaluate	10.0	Changes		
11	Test Market	15.0	Changes		
12	Changes	5.0	Production		
13	Production	0.0			
14	Marketing	0.0			

The same Holiday data set is used. To indicate that work is to be done on all days of the week except Sunday, use INTERVAL=DTDAY and define a Calendar data set with a single variable _SUN_, and a single observation identifying Sunday as a holiday. The DATA step creating CALENDAR and the invocation of PROC CPM is shown in the following code. Output 4.9.2 displays the resulting schedule.

```
 /* Set up a 6-day work week, with Sundays off */
 data calendar;
    _sun_='holiday';
    run;

title 'Scheduling on the 6-Day Week';
proc cpm data=widget9 holidata=holidays
         out=savec date='1dec03:07:00'dt
         interval=dtday daylength='08:30't
         calendar=calendar;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   run;
```

Output 4.9.2 Scheduling on the Six-Day Week

Scheduling on the 6-Day Week
Project Schedule

Obs	task	days	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	Approve Plan	5.5	01DEC03:07:00:00	06DEC03:11:14:59	01DEC03:07:00:00	06DEC03:11:14:59	0.0	0.0
2	Drawings	10.0	06DEC03:11:15:00	18DEC03:11:14:59	06DEC03:11:15:00	18DEC03:11:14:59	0.0	0.0
3	Study Market	5.0	06DEC03:11:15:00	12DEC03:11:14:59	15JAN04:11:15:00	21JAN04:11:14:59	30.0	0.0
4	Write Specs	4.5	06DEC03:11:15:00	11DEC03:15:29:59	13DEC03:07:00:00	18DEC03:11:14:59	5.5	5.5
5	Prototype	15.0	18DEC03:11:15:00	09JAN04:11:14:59	18DEC03:11:15:00	09JAN04:11:14:59	0.0	0.0
6	Mkt. Strat.	10.0	12DEC03:11:15:00	27DEC03:11:14:59	21JAN04:11:15:00	02FEB04:11:14:59	30.0	30.0
7	Materials	10.0	09JAN04:11:15:00	21JAN04:11:14:59	09JAN04:11:15:00	21JAN04:11:14:59	0.0	0.0
8	Facility	10.0	09JAN04:11:15:00	21JAN04:11:14:59	09JAN04:11:15:00	21JAN04:11:14:59	0.0	0.0
9	Init. Prod.	10.0	21JAN04:11:15:00	02FEB04:11:14:59	21JAN04:11:15:00	02FEB04:11:14:59	0.0	0.0
10	Evaluate	10.0	02FEB04:11:15:00	13FEB04:11:14:59	07FEB04:11:15:00	19FEB04:11:14:59	5.0	5.0
11	Test Market	15.0	02FEB04:11:15:00	19FEB04:11:14:59	02FEB04:11:15:00	19FEB04:11:14:59	0.0	0.0
12	Changes	5.0	19FEB04:11:15:00	25FEB04:11:14:59	19FEB04:11:15:00	25FEB04:11:14:59	0.0	0.0
13	Production	0.0	25FEB04:11:15:00	25FEB04:11:15:00	25FEB04:11:15:00	25FEB04:11:15:00	0.0	0.0
14	Marketing	0.0	02FEB04:11:15:00	02FEB04:11:15:00	25FEB04:11:15:00	25FEB04:11:15:00	20.0	20.0

Suppose now that you want to schedule work on a five-and-a-half day week (five full working days starting on Monday and half a working day on Saturday). A full work day is from 8 a.m. to 4 p.m. Output 4.9.3 shows the data set WORKDAT, which is used to define the work pattern for a full day (in the shift variable fullday and a half-day (in the shift variable halfday). Output 4.9.4 displays the Calendar data set, CALDAT, which specifies the appropriate work pattern for each day of the week. The schedule produced by invoking the following program is displayed in Output 4.9.5.

Output 4.9.3 Workday Data Set

Scheduling on a Five-and-a-Half-Day Week
Workdays Data Set

Obs	fullday	halfday
1	8:00	8:00
2	16:00	12:00

Output 4.9.4 Calendar Data Set

Scheduling on a Five-and-a-Half-Day Week
Calendar Data Set

Obs	_sun_	_mon_	_tue_	_wed_	_thu_	_fri_	_sat_	d_length
1	holiday	fullday	fullday	fullday	fullday	fullday	halfday	8:00

```
proc cpm data=widget9 holidata=holidays
         out=savecw date='1dec03'd
         interval=day
         workday=workdat calendar=caldat;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   run;
```

Output 4.9.5 Scheduling on a Five-and-a-Half Day Week

Scheduling on a Five-and-a-Half-Day Week
Project Schedule

Obs	task	days	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	Approve Plan	5.5	01DEC03:08:00:00	06DEC03:11:59:59	01DEC03:08:00:00	06DEC03:11:59:59	0.0	0.0
2	Drawings	10.0	08DEC03:08:00:00	19DEC03:11:59:59	08DEC03:08:00:00	19DEC03:11:59:59	0.0	0.0
3	Study Market	5.0	08DEC03:08:00:00	12DEC03:15:59:59	20JAN04:08:00:00	26JAN04:11:59:59	30.0	0.0
4	Write Specs	4.5	08DEC03:08:00:00	12DEC03:11:59:59	15DEC03:08:00:00	19DEC03:11:59:59	5.5	5.5
5	Prototype	15.0	19DEC03:12:00:00	13JAN04:11:59:59	19DEC03:12:00:00	13JAN04:11:59:59	0.0	0.0
6	Mkt. Strat.	10.0	13DEC03:08:00:00	30DEC03:11:59:59	26JAN04:12:00:00	06FEB04:15:59:59	30.0	30.0
7	Materials	10.0	13JAN04:12:00:00	26JAN04:11:59:59	13JAN04:12:00:00	26JAN04:11:59:59	0.0	0.0
8	Facility	10.0	13JAN04:12:00:00	26JAN04:11:59:59	13JAN04:12:00:00	26JAN04:11:59:59	0.0	0.0
9	Init. Prod.	10.0	26JAN04:12:00:00	06FEB04:15:59:59	26JAN04:12:00:00	06FEB04:15:59:59	0.0	0.0
10	Evaluate	10.0	07FEB04:08:00:00	19FEB04:15:59:59	13FEB04:12:00:00	26FEB04:11:59:59	5.0	5.0
11	Test Market	15.0	07FEB04:08:00:00	26FEB04:11:59:59	07FEB04:08:00:00	26FEB04:11:59:59	0.0	0.0
12	Changes	5.0	26FEB04:12:00:00	03MAR04:15:59:59	26FEB04:12:00:00	03MAR04:15:59:59	0.0	0.0
13	Production	0.0	04MAR04:08:00:00	04MAR04:08:00:00	04MAR04:08:00:00	04MAR04:08:00:00	0.0	0.0
14	Marketing	0.0	07FEB04:08:00:00	07FEB04:08:00:00	04MAR04:08:00:00	04MAR04:08:00:00	20.0	20.0

Note that, in this case, it was not necessary to specify the DAYLENGTH=, DAYSTART=, or INTERVAL= option in the PROC CPM statement. The default value of INTERVAL=DAY is assumed, and the CALDAT and WORKDAT data sets define the workday and work week completely. The length of a standard working day is also included in the Calendar data set, completing all the necessary specifications.

To visualize the breaks in the work schedule created by these specifications, you can use the following simple data set with a dummy activity 'Schedule Breaks' to produce a Gantt chart, shown in Output 4.9.6. The period illustrated on the chart is from December 19, 2003 to December 27, 2003. The breaks are denoted by *.

```
/* To visualize the breaks, use following "dummy" data set
   to plot a schedule bar showing holidays and breaks */
data temp;
   e_start='19dec03:08:00'dt;
   e_finish='27dec03:23:59:59'dt;
   task='Schedule Breaks';
   label task='Project Calendar';
   format e_start e_finish datetime16.;
   run;
```

```
title2 'Holidays and Breaks in the Project Calendar';
proc gantt data=temp lineprinter
           calendar=caldat holidata=holidays
           workday=workdat;
   chart / interval=dtday mininterval=dthour skip=0
           holiday=(holiday) holifin=(holifin) markbreak
           nojobnum nolegend increment=8 holichar='*';
   id task;
   run;
```

Output 4.9.6 Gantt Chart Showing Breaks and Holidays

Scheduling on a Five-and-a-Half-Day Week
Holidays and Breaks in the Project Calendar

```
                   DEC     DEC     DEC     DEC     DEC     DEC
Project            19      19      20      20      20      21
Calendar           08:00   16:00   00:00   08:00   16:00   00:00
                  -+-------+-------+-------+-------+-------+-
Schedule Breaks   |<-------****************----*************|
                  -+-------+-------+-------+-------+-------+-
```

Scheduling on a Five-and-a-Half-Day Week
Holidays and Breaks in the Project Calendar

```
 DEC     DEC     DEC     DEC     DEC     DEC     DEC     DEC
 21      21      21      22      22      22      23      23
 00:00   08:00   16:00   00:00   08:00   16:00   00:00   08:00
-+-------+-------+-------+-------+-------+-------+-------+-
|********************************--------****************-|
-+-------+-------+-------+-------+-------+-------+-------+-
```

Scheduling on a Five-and-a-Half-Day Week
Holidays and Breaks in the Project Calendar

```
 DEC     DEC     DEC     DEC     DEC     DEC     DEC     DEC
 23      23      24      24      24      25      25      25
 08:00   16:00   00:00   08:00   16:00   00:00   08:00   16:00
-+-------+-------+-------+-------+-------+-------+-------+-
|--------*************************************************|
-+-------+-------+-------+-------+-------+-------+-------+-
```

Scheduling on a Five-and-a-Half-Day Week
Holidays and Breaks in the Project Calendar

```
 DEC     DEC     DEC     DEC     DEC     DEC     DEC     DEC
 25      26      26      26      27      27      27      28
 16:00   00:00   08:00   16:00   00:00   08:00   16:00   00:00
-+-------+-------+-------+-------+-------+-------+-------+-
|****************************************----*************|
-+-------+-------+-------+-------+-------+-------+-------+-
```

Example 4.10: Multiple Calendars

This example illustrates the use of multiple calendars within a project. Different scenarios are presented to show the use of different calendars and how project schedules are affected. Output 4.10.1 shows the data set WORKDATA, which defines several shift patterns. These shift patterns are appropriately associated with three different calendars in the data set CALEDATA, also shown in the same output. The three calendars are defined as follows:

- The DEFAULT calendar has five eight-hour days (Monday through Friday) and holidays on Saturday and Sunday.
- The calendar OVT_CAL specifies an overtime calendar that has 10-hour work days on Monday through Friday and a half day on Saturday and a holiday on Sunday.
- The calendar PROD_CAL follows a more complicated work pattern: Sunday is a holiday; on Monday work is done from 8 a.m. through midnight with a two hour break from 6 p.m. to 8 p.m.; on Tuesday through Friday work is done round the clock with two 2-hour breaks from 6 a.m. to 8 a.m. and 6 p.m. to 8 p.m.; on Saturday the work shifts are from midnight to 6 a.m. and again from 8 a.m. to 6 p.m. In other words, work is done continuously from 8 a.m. on Monday morning to 6 p.m. on Saturday with two hour breaks every day at 6 a.m. and 6 p.m.

Output 4.10.1 Workday and Calendar Data Sets

Multiple Calendars
Workdays Data Set

Obs	fullday	halfday	ovtday	s1	s2	s3
1	8:00	8:00	8:00	.	8:00	.
2	16:00	12:00	18:00	6:00	18:00	6:00
3	.	.	.	8:00	20:00	8:00
4	.	.	.	18:00	.	18:00
5	.	.	.	20:00	.	.
6	.	.	.	.	.	.

Multiple Calendars
CALENDAR Data Set

Obs	cal	_sun_	_mon_	_tue_	_wed_	_thu_	_fri_	_sat_
1	DEFAULT	holiday	fullday	fullday	fullday	fullday	fullday	holiday
2	OVT_CAL	holiday	ovtday	ovtday	ovtday	ovtday	ovtday	halfday
3	PROD_CAL	holiday	s2	s1	s1	s1	s1	s3

The same set of holidays is used as in Example 4.9, except that in this case the holiday for New Year's is defined by specifying both the start and finish time for the holiday instead of defaulting to a one-day long holiday. When multiple calendars are involved, it is often less confusing to define holidays by specifying both a start and a finish time for the holiday instead of the start time and duration. Output 4.10.2 displays the Holiday data set.

Output 4.10.2 Holiday Data Set

Multiple Calendars
Holidays Data Set

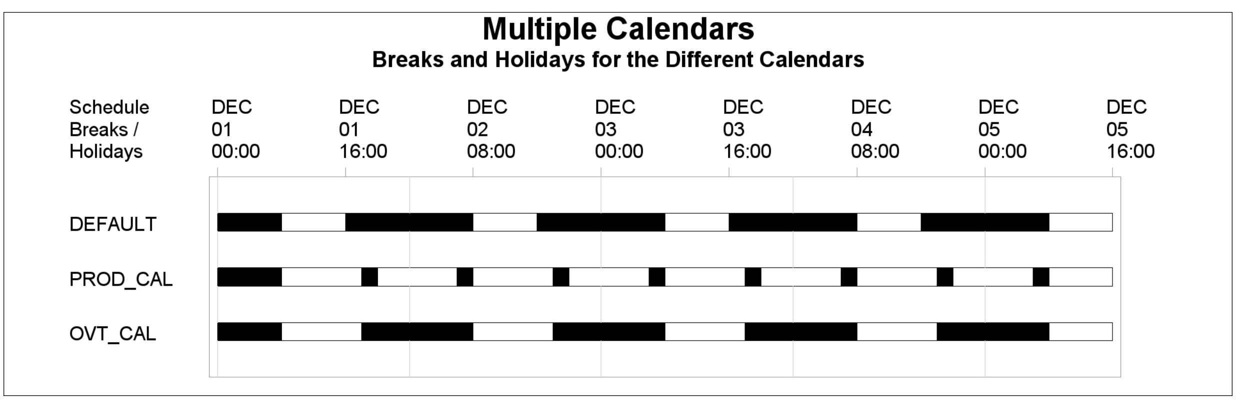
Obs	holiday	holifin	holidur
1	24DEC03	26DEC03	4
2	01JAN04	01JAN04	.

The data set HOLIDAYS does not include any variable identifying the calendars with which to associate the holidays. By default, the procedure associates the two holiday periods with all the calendars.

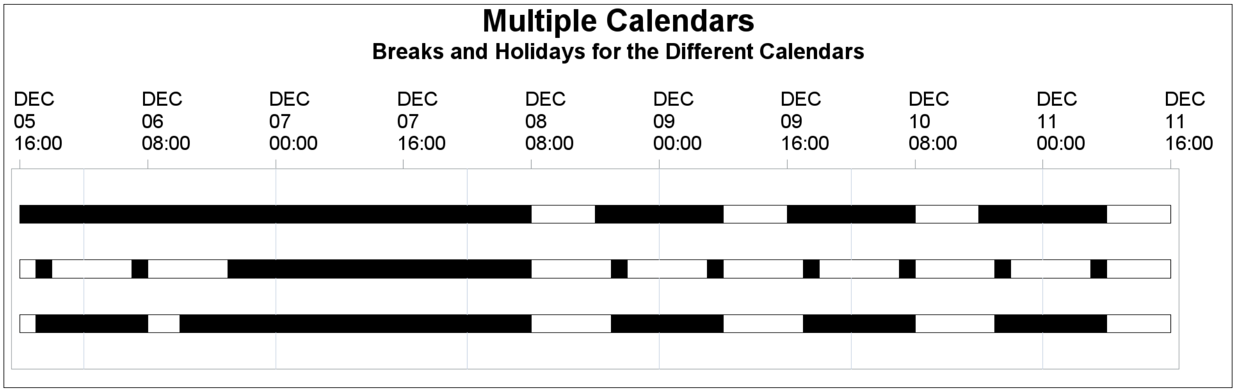
An easy way to visualize all the breaks and holidays for each calendar is to use a Gantt chart, plotting a bar for each calendar from the start of the project to January 4, 2004, with all the holiday and work shift specifications. The following program produces Output 4.10.3. Holidays and breaks are marked with a solid fill pattern.

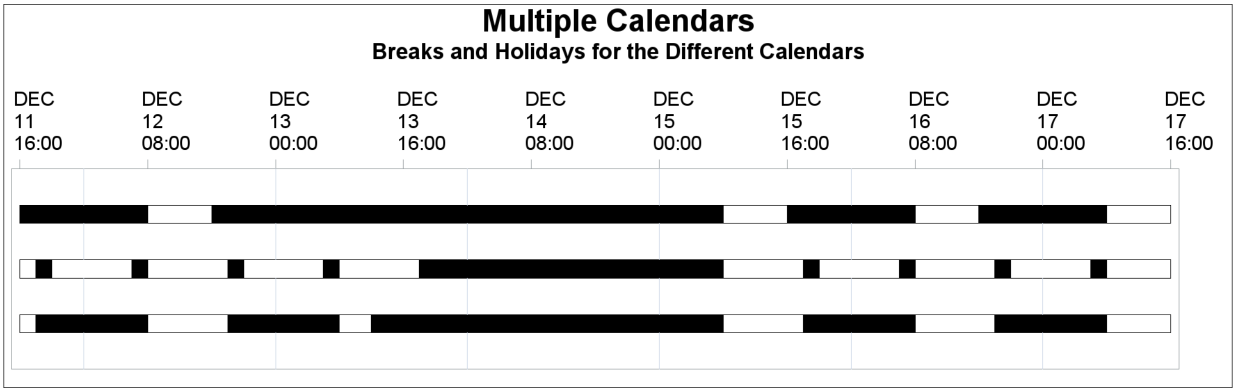
```
goptions hpos=160 vpos=25;
title h=2 'Multiple Calendars';
title2 h=1.4 'Breaks and Holidays for the Different Calendars';
proc gantt data=cals graphics
           calendar=calendar holidata=holidays
           workday=workdata;
   chart / interval=dtday mininterval=dthour skip=2
           holiday=(holiday) holifin=(holifin)
           markbreak daylength='08:00't calid=cal
           ref='1dec03:00:00'dt to '4jan04:08:00'dt by dtday
           nolegend nojobnum increment=16
           hpages=6;
   id cal;
   run;
```

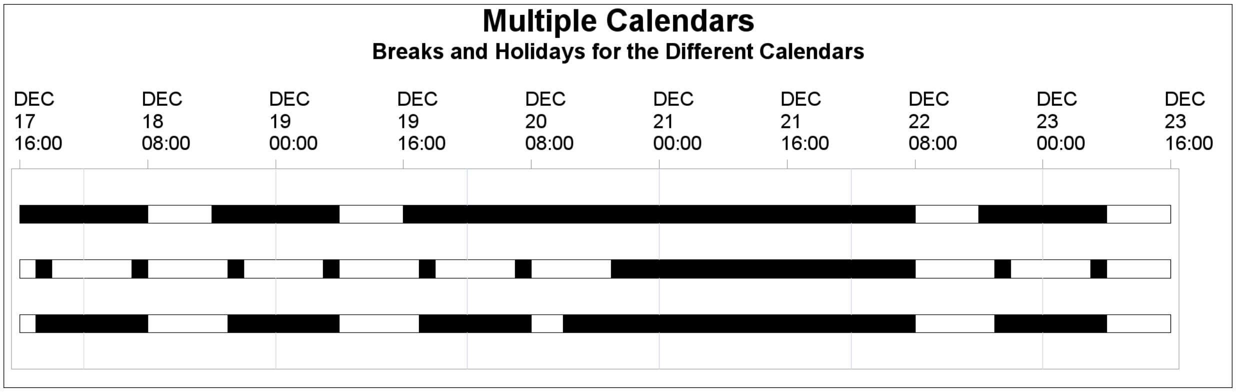
Output 4.10.3 Gantt Chart Showing Breaks and Holidays for Multiple Calendars

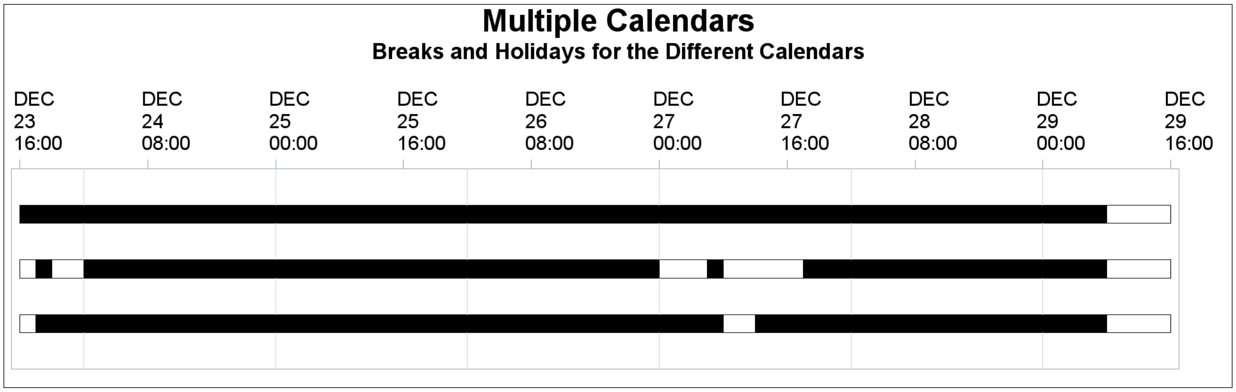


Output 4.10.3 *continued*

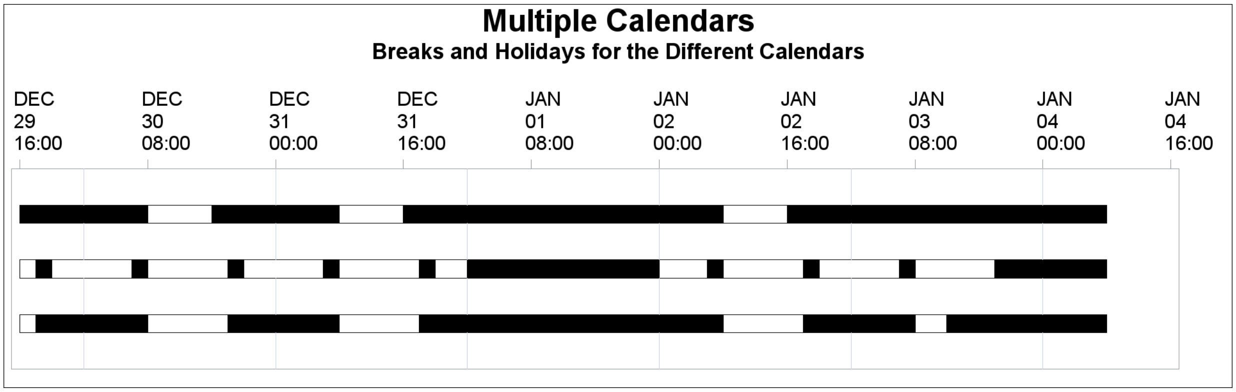








Output 4.10.3 *continued*



The Activity data set used in Example 4.9 is modified by adding a variable called cal, which sets the calendar to be 'PROD_CAL' for the activity 'Production', and 'OVT_CAL' for the activity 'Prototype', and the DEFAULT calendar for the other activities. Thus, in both the Activity data set and the Calendar data set, the calendar information is conveyed through a CALID variable, cal.

PROC CPM is first invoked without reference to the CALID variable. Thus, the procedure recognizes only the first observation in the Calendar data set (a warning is printed to the log to this effect), and only the default calendar is used for all activities in the project. The daylength parameter is interpreted as the length of a standard work day; all the durations are assumed to be in units of this standard work day. Output 4.10.4 displays the schedule obtained. The project is scheduled to finish on March 12, 2004, at 12 noon.

```
data widgcal;
   set widget9;
   if task = 'Production' then       cal = 'PROD_CAL';
   else if task = 'Prototype' then   cal = 'OVT_CAL';
   else                              cal = 'DEFAULT';
   run;

proc cpm date='01dec03'd data=widgcal out=scheddef
         holidata=holidays daylength='08:00't
         workday=workdata
         calendar=calendar;
   holiday holiday / holifin = holifin;
   activity task;
   duration days;
   successor succ1 succ2 succ3;
   run;

title2 'Project Schedule: Default calendar';
proc print heading=h;
   var task days e_start e_finish l_start l_finish
       t_float f_float;
   run;
```

Output 4.10.4 Schedule Using Default Calendar

Multiple Calendars
Project Schedule: Default calendar

Obs	task	days	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	Approve Plan	5.5	01DEC03:08:00:00	08DEC03:11:59:59	01DEC03:08:00:00	08DEC03:11:59:59	0.0	0.0
2	Drawings	10.0	08DEC03:12:00:00	22DEC03:11:59:59	08DEC03:12:00:00	22DEC03:11:59:59	0.0	0.0
3	Study Market	5.0	08DEC03:12:00:00	15DEC03:11:59:59	23JAN04:12:00:00	30JAN04:11:59:59	30.0	0.0
4	Write Specs	4.5	08DEC03:12:00:00	12DEC03:15:59:59	16DEC03:08:00:00	22DEC03:11:59:59	5.5	5.5
5	Prototype	15.0	22DEC03:12:00:00	16JAN04:11:59:59	22DEC03:12:00:00	16JAN04:11:59:59	0.0	0.0
6	Mkt. Strat.	10.0	15DEC03:12:00:00	02JAN04:11:59:59	30JAN04:12:00:00	13FEB04:11:59:59	30.0	30.0
7	Materials	10.0	16JAN04:12:00:00	30JAN04:11:59:59	16JAN04:12:00:00	30JAN04:11:59:59	0.0	0.0
8	Facility	10.0	16JAN04:12:00:00	30JAN04:11:59:59	16JAN04:12:00:00	30JAN04:11:59:59	0.0	0.0
9	Init. Prod.	10.0	30JAN04:12:00:00	13FEB04:11:59:59	30JAN04:12:00:00	13FEB04:11:59:59	0.0	0.0
10	Evaluate	10.0	13FEB04:12:00:00	27FEB04:11:59:59	20FEB04:12:00:00	05MAR04:11:59:59	5.0	5.0
11	Test Market	15.0	13FEB04:12:00:00	05MAR04:11:59:59	13FEB04:12:00:00	05MAR04:11:59:59	0.0	0.0
12	Changes	5.0	05MAR04:12:00:00	12MAR04:11:59:59	05MAR04:12:00:00	12MAR04:11:59:59	0.0	0.0
13	Production	0.0	12MAR04:12:00:00	12MAR04:12:00:00	12MAR04:12:00:00	12MAR04:12:00:00	0.0	0.0
14	Marketing	0.0	13FEB04:12:00:00	13FEB04:12:00:00	12MAR04:12:00:00	12MAR04:12:00:00	20.0	20.0

Next PROC CPM is invoked with the CALID statement identifying the variable CAL in the Activity and Calendar data sets. Recall that the two activities, ‘Production’ and ‘Prototype’, do not follow the default calendar. The schedule displayed in Output 4.10.5 shows that, due to longer working hours for these two activities in the project, the scheduled finish date is now March 8, at 10:00 a.m.

```
proc cpm date='01dec03'd data=widgcal out=schedmc
         holidata=holidays daylength='08:00't
         workday=workdata
         calendar=calendar;
   holiday holiday / holifin = holifin;
   activity task;
   duration days;
   successor succ1 succ2 succ3;
   calid cal;
   run;
title2 'Project Schedule: Three Calendars';
proc print;
   var task days cal e_: l_: t_float f_float;
   run;
```

Output 4.10.5 Schedule Using Three Calendars

Multiple Calendars
Project Schedule: Three Calendars

Obs	task	days	cal	E_START	E_FINISH	L_START
1	Approve Plan	5.5	DEFAULT	01DEC03:08:00:00	08DEC03:11:59:59	01DEC03:08:00:00
2	Drawings	10.0	DEFAULT	08DEC03:12:00:00	22DEC03:11:59:59	08DEC03:12:00:00
3	Study Market	5.0	DEFAULT	08DEC03:12:00:00	15DEC03:11:59:59	19JAN04:10:00:00
4	Write Specs	4.5	DEFAULT	08DEC03:12:00:00	12DEC03:15:59:59	16DEC03:08:00:00
5	Prototype	15.0	OVT_CAL	22DEC03:12:00:00	12JAN04:09:59:59	22DEC03:12:00:00
6	Mkt. Strat.	10.0	DEFAULT	15DEC03:12:00:00	02JAN04:11:59:59	26JAN04:10:00:00
7	Materials	10.0	DEFAULT	12JAN04:10:00:00	26JAN04:09:59:59	12JAN04:10:00:00
8	Facility	10.0	DEFAULT	12JAN04:10:00:00	26JAN04:09:59:59	12JAN04:10:00:00
9	Init. Prod.	10.0	DEFAULT	26JAN04:10:00:00	09FEB04:09:59:59	26JAN04:10:00:00
10	Evaluate	10.0	DEFAULT	09FEB04:10:00:00	23FEB04:09:59:59	16FEB04:10:00:00
11	Test Market	15.0	DEFAULT	09FEB04:10:00:00	01MAR04:09:59:59	09FEB04:10:00:00
12	Changes	5.0	DEFAULT	01MAR04:10:00:00	08MAR04:09:59:59	01MAR04:10:00:00
13	Production	0.0	PROD_CAL	08MAR04:10:00:00	08MAR04:10:00:00	08MAR04:10:00:00
14	Marketing	0.0	DEFAULT	09FEB04:10:00:00	09FEB04:10:00:00	08MAR04:10:00:00

Obs	L_FINISH	T_FLOAT	F_FLOAT
1	08DEC03:11:59:59	0.00	0.00
2	22DEC03:11:59:59	0.00	0.00
3	26JAN04:09:59:59	25.75	0.00
4	22DEC03:11:59:59	5.50	5.50
5	12JAN04:09:59:59	0.00	0.00
6	09FEB04:09:59:59	25.75	25.75
7	26JAN04:09:59:59	0.00	0.00
8	26JAN04:09:59:59	0.00	0.00
9	09FEB04:09:59:59	0.00	0.00
10	01MAR04:09:59:59	5.00	5.00
11	01MAR04:09:59:59	0.00	0.00
12	08MAR04:09:59:59	0.00	0.00
13	08MAR04:10:00:00	0.00	0.00
14	08MAR04:10:00:00	20.00	20.00

Now suppose that the engineer in charge of writing specifications requests a seven-day vacation from December 8, 2003. How is the project completion time going to be affected? A new calendar, Eng_cal, is defined that has the same work pattern as the default calendar, but it also contains an extra vacation period. Output 4.10.6 displays the data sets HOLIDATA and CALEDATA, which contain information about the new calendar. The fourth observation in the data set CALEDATA has missing values for the variables _sun_, . . . , _sat_, indicating that the calendar, Eng_cal, follows the same work pattern as the default calendar.

Output 4.10.6 HOLIDATA and CALEDATA Data Sets

Multiple Calendars
Holidays Data Set

Obs	holiday	holifin	holidur	cal
1	08DEC03	.	7	Eng_cal
2	24DEC03	26DEC03	.	
3	01JAN04	01JAN04	.	

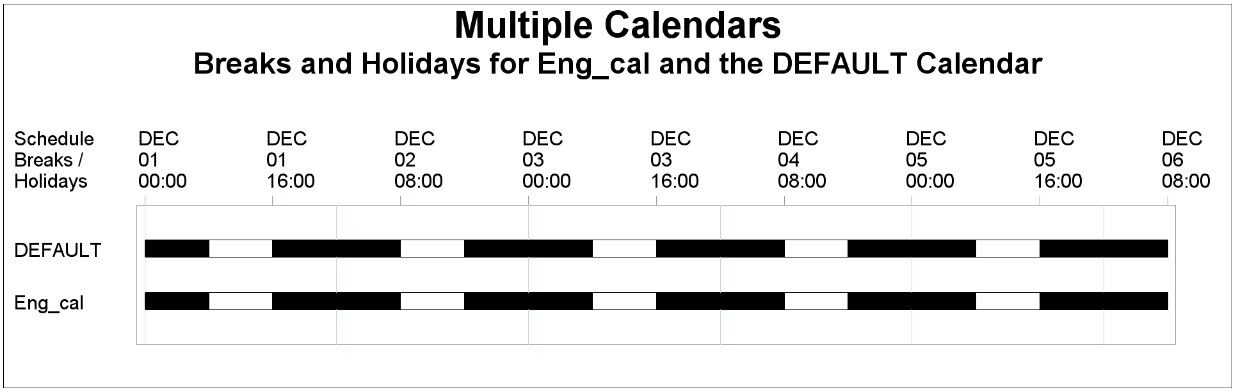
Multiple Calendars
Calendar Data Set

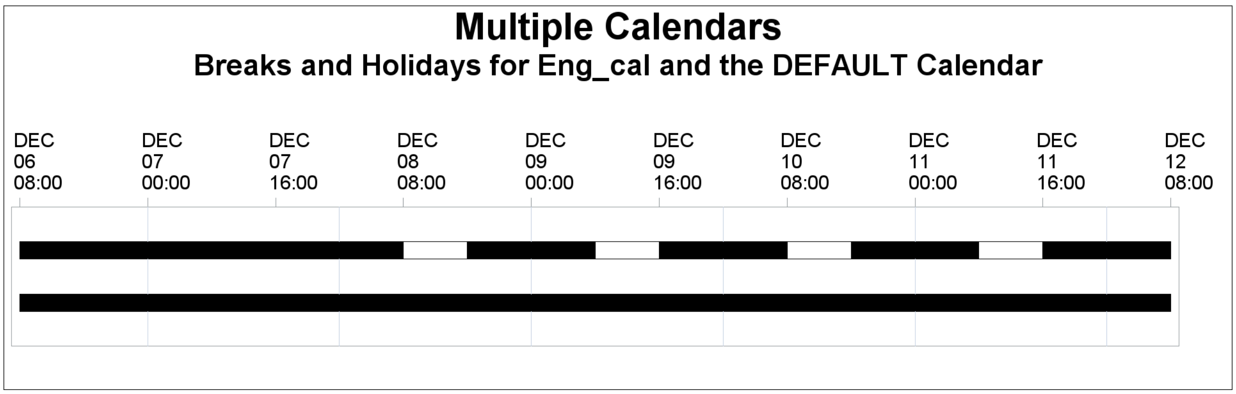
Obs	cal	_sun_	_mon_	_tue_	_wed_	_thu_	_fri_	_sat_
1	DEFAULT	holiday	fullday	fullday	fullday	fullday	fullday	holiday
2	OVT_CAL	holiday	ovtday	ovtday	ovtday	ovtday	ovtday	halfday
3	PROD_CAL	holiday	s2	s1	s1	s1	s1	s3
4	Eng_cal							

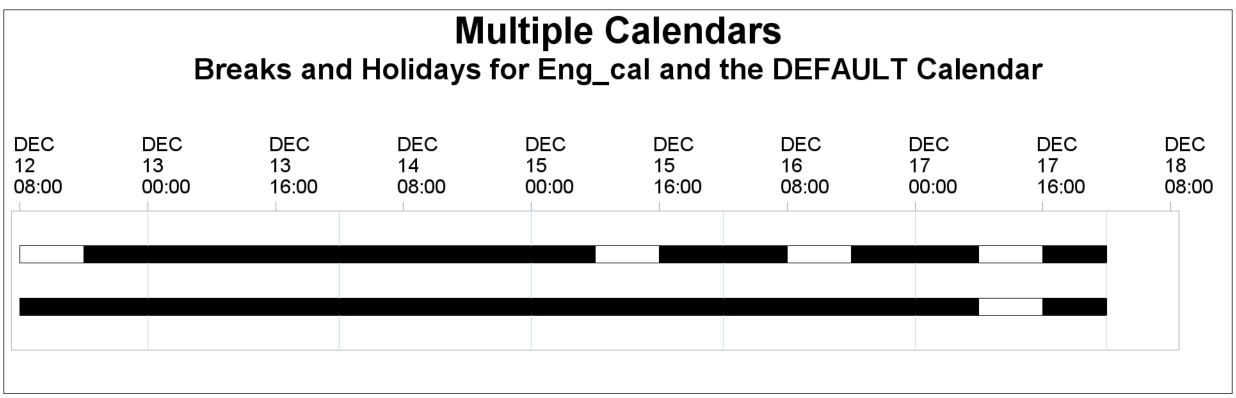
Once again, in the following code, PROC GANTT is used to compare the new calendar with the default calendar, as shown in Output 4.10.7. Note that the breaks and holidays are marked with a solid fill pattern.

```
/* Create a data set to illustrate holidays with PROC GANTT */
data cals2;
   e_start='1dec03:00:00'dt;
   e_finish='18dec03:00:00'dt;
   label cal ='Schedule Breaks / Holidays';
   format e_start e_finish datetime16.;
   length cal $8.;
   cal='DEFAULT' ; output;
   cal='Eng_cal' ; output;
   run;

title2 'Breaks and Holidays for Eng_cal and the DEFAULT Calendar';
proc gantt data=cals2 graphics
           calendar=caledata holidata=holidata
           workday=workdata;
   chart / interval=dtday mininterval=dthour skip=2
           holiday=(holiday) holifin=(holifin) holidur=(holidur)
           markbreak daylength='08:00't calid=cal
           ref='1dec03:00:00'dt to '18dec03:00:00'dt by dtday
           nojobnum nolegend increment=16 hpages=3;
   id cal;
   run;
```

Output 4.10.7 Difference between Eng_cal and DEFAULT Calendar







The Activity data set is modified to redefine the calendar for the task 'Write Specs'. PROC CPM is invoked, and Output 4.10.8 shows the new schedule obtained. Note the effect of the Engineer's vacation on the project completion time. The project is now scheduled to finish at 10 a.m. on March 9, 2004; in effect, the delay is only one day, even though the planned vacation period is seven days. This is due to the fact that the activity 'Write Specs', which follows the new calendar, had some slack time present in its original schedule; however, this activity has now become critical.

```
data widgvac;
   set widgcal;
   if task = 'Write Specs' then cal = 'Eng_cal';
   run;
```

```
proc cpm date='01dec03'd data=widgvac out=schedvac
         holidata=holidata daylength='08:00't
         workday=workdata
         calendar=caledata;
   holiday holiday / holifin = holifin holidur=holidur;
   activity task;
   duration days;
   successor succ1 succ2 succ3;
   calid cal;
   run;

title2 'Project Schedule: Four Calendars';
proc print;
   var task days cal e_: l_: t_float f_float;
   run;
```

Output 4.10.8 Schedule Using Four Calendars

Multiple Calendars
Project Schedule: Four Calendars

Obs	task	days	cal	E_START	E_FINISH	L_START
1	Approve Plan	5.5	DEFAULT	01DEC03:08:00:00	08DEC03:11:59:59	02DEC03:08:00:00
2	Drawings	10.0	DEFAULT	08DEC03:12:00:00	22DEC03:11:59:59	09DEC03:12:00:00
3	Study Market	5.0	DEFAULT	08DEC03:12:00:00	15DEC03:11:59:59	20JAN04:10:00:00
4	Write Specs	4.5	Eng_cal	17DEC03:08:00:00	23DEC03:11:59:59	17DEC03:08:00:00
5	Prototype	15.0	OVT_CAL	23DEC03:12:00:00	13JAN04:09:59:59	23DEC03:12:00:00
6	Mkt. Strat.	10.0	DEFAULT	15DEC03:12:00:00	02JAN04:11:59:59	27JAN04:10:00:00
7	Materials	10.0	DEFAULT	13JAN04:10:00:00	27JAN04:09:59:59	13JAN04:10:00:00
8	Facility	10.0	DEFAULT	13JAN04:10:00:00	27JAN04:09:59:59	13JAN04:10:00:00
9	Init. Prod.	10.0	DEFAULT	27JAN04:10:00:00	10FEB04:09:59:59	27JAN04:10:00:00
10	Evaluate	10.0	DEFAULT	10FEB04:10:00:00	24FEB04:09:59:59	17FEB04:10:00:00
11	Test Market	15.0	DEFAULT	10FEB04:10:00:00	02MAR04:09:59:59	10FEB04:10:00:00
12	Changes	5.0	DEFAULT	02MAR04:10:00:00	09MAR04:09:59:59	02MAR04:10:00:00
13	Production	0.0	PROD_CAL	09MAR04:10:00:00	09MAR04:10:00:00	09MAR04:10:00:00
14	Marketing	0.0	DEFAULT	10FEB04:10:00:00	10FEB04:10:00:00	09MAR04:10:00:00

Obs	L_FINISH	T_FLOAT	F_FLOAT
1	09DEC03:11:59:59	1.00	0.00
2	23DEC03:11:59:59	1.00	1.00
3	27JAN04:09:59:59	26.75	0.00
4	23DEC03:11:59:59	0.00	0.00
5	13JAN04:09:59:59	0.00	0.00
6	10FEB04:09:59:59	26.75	26.75
7	27JAN04:09:59:59	0.00	0.00
8	27JAN04:09:59:59	0.00	0.00
9	10FEB04:09:59:59	0.00	0.00
10	02MAR04:09:59:59	5.00	5.00
11	02MAR04:09:59:59	0.00	0.00
12	09MAR04:09:59:59	0.00	0.00
13	09MAR04:10:00:00	0.00	0.00
14	09MAR04:10:00:00	20.00	20.00

Example 4.11: Nonstandard Relationships

This example shows the use of LAG variables to describe nonstandard relationships. Consider the project network in AON format. Output 4.11.1 shows the data set WIDGLAG, which contains the required project information; here the data set contains only one successor variable, requiring multiple observations for activities that have more than one immediate successor. In addition, the data set contains two new variables, lagdur and lagdurc, which are used to convey nonstandard relationships that exist between some of the activities. In the first part of the example, lagdur specifies a lag type and lag duration between activities; in the second part, the variable lagdurc specifies a lag calendar in addition to the lag type and lag duration. When multiple successor variables are used, you can specify multiple lag variables and the lag values specified are matched one-for-one with the corresponding successor variables.

Output 4.11.1 Network Data

Non-Standard Relationships
Activity Data Set WIDGLAG

Obs	task	days	succ	lagdur	lagdurc
1	Approve Plan	5	Drawings		
2	Approve Plan	5	Study Market		
3	Approve Plan	5	Write Specs		
4	Drawings	10	Prototype		
5	Study Market	5	Mkt. Strat.		
6	Write Specs	5	Prototype		
7	Prototype	15	Materials	ss_9	ss_9
8	Prototype	15	Facility	ss_9	ss_9
9	Mkt. Strat.	10	Test Market		
10	Mkt. Strat.	10	Marketing		
11	Materials	10	Init. Prod.		
12	Facility	10	Init. Prod.	fs_2	fs_2_SEVENDAY
13	Init. Prod.	10	Test Market		
14	Init. Prod.	10	Marketing		
15	Init. Prod.	10	Evaluate		
16	Evaluate	10	Changes		
17	Test Market	15	Changes		
18	Changes	5	Production		
19	Production	0			
20	Marketing	0			

Suppose that the project calendar follows a five-day work week. Recall from Example 4.6 that the project finishes on March 8, 2004. The data set, WIDGLAG, specifies that there is a 'ss_9' lag between the activities 'Prototype' and 'Materials', which means that you can start acquiring raw materials nine days after the start of the activity 'Prototype' instead of waiting until its finish time. Likewise, there is an 'ss_9' lag between 'Prototype' and 'Facility'. The 'fs_2' lag between 'Facility' and 'Init. Prod' indicates that you should wait two days after the completion of the 'Facility' task before starting the initial production. To convey the lag information to PROC CPM, use the LAG= specification in the SUCCESSOR statement. The program and the resulting output (Output 4.11.2) follow.

```
proc cpm data=widglag date='1dec03'd
         interval=weekday collapse out=lagsched;
   activity task;
   succ     succ / lag = (lagdur);
   duration days;
   run;
```

Output 4.11.2 Project Schedule: Default LAG Calendar

Non-Standard Relationships
Lag Type and Duration: Default LAG Calendar

task	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Approve Plan	01DEC03	05DEC03	01DEC03	05DEC03	0	0
Drawings	08DEC03	19DEC03	08DEC03	19DEC03	0	0
Study Market	08DEC03	12DEC03	13JAN04	19JAN04	26	0
Write Specs	08DEC03	12DEC03	15DEC03	19DEC03	5	5
Prototype	22DEC03	09JAN04	22DEC03	09JAN04	0	0
Mkt. Strat.	15DEC03	26DEC03	20JAN04	02FEB04	26	26
Materials	02JAN04	15JAN04	06JAN04	19JAN04	2	2
Facility	02JAN04	15JAN04	02JAN04	15JAN04	0	0
Init. Prod.	20JAN04	02FEB04	20JAN04	02FEB04	0	0
Evaluate	03FEB04	16FEB04	10FEB04	23FEB04	5	5
Test Market	03FEB04	23FEB04	03FEB04	23FEB04	0	0
Changes	24FEB04	01MAR04	24FEB04	01MAR04	0	0
Production	02MAR04	02MAR04	02MAR04	02MAR04	0	0
Marketing	03FEB04	03FEB04	02MAR04	02MAR04	20	20

Due to the change in the type of precedence constraint (from the default 'fs_0' to 'ss_9'), the project finishes earlier, on March 2, 2004, instead of on March 8, 2004 (compare with Output 4.6.1).

By default, all the lags are assumed to follow the default calendar for the project. In this case, the default project calendar has five workdays (since INTERVAL=WEEKDAY). Suppose now that the 'fs_2' lag between 'Facility' and 'Init. Prod.' really indicates two calendar days and not two workdays. (Perhaps you want to allow two days for the paint to dry or the building to be ventilated.) The variable lagdurc in the WIDGLAG data set indicates the calendar for this lag by specifying the lag to be 'fs_2_sevenday' where 'sevenday' is the name of the seven-day calendar defined in the Calendar data set, CALENDAR, displayed in Output 4.11.3. PROC CPM is invoked with LAG=lagdurc and Output 4.11.4 displays the resulting schedule. Note that the project now finishes on March 1, 2004.

Output 4.11.3 Calendar Data Set

Non-Standard Relationships
Calendar Data Set

Obs	_cal_	_sun_	_mon_	_tue_	_wed_	_thu_	_fri_	_sat_
1	SEVENDAY	workday	workday	workday	workday	workday	workday	workday

```
proc cpm data=widglag date='1dec03'd calendar=calendar
         interval=weekday collapse out=lagsched;
   activity task;
   succ     succ / lag = (lagdurc);
   duration days;
   run;
```

Output 4.11.4 Project Schedule: Lag Type, Duration, and Calendar

Non-Standard Relationships
Lag Type, Duration, and Calendar

task	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Approve Plan	01DEC03	05DEC03	02DEC03	08DEC03	1	0
Drawings	08DEC03	19DEC03	09DEC03	22DEC03	1	0
Study Market	08DEC03	12DEC03	12JAN04	16JAN04	25	0
Write Specs	08DEC03	12DEC03	16DEC03	22DEC03	6	5
Prototype	22DEC03	09JAN04	23DEC03	12JAN04	1	0
Mkt. Strat.	15DEC03	26DEC03	19JAN04	30JAN04	25	25
Materials	02JAN04	15JAN04	05JAN04	16JAN04	1	1
Facility	02JAN04	15JAN04	05JAN04	16JAN04	1	1
Init. Prod.	19JAN04	30JAN04	19JAN04	30JAN04	0	0
Evaluate	02FEB04	13FEB04	09FEB04	20FEB04	5	5
Test Market	02FEB04	20FEB04	02FEB04	20FEB04	0	0
Changes	23FEB04	27FEB04	23FEB04	27FEB04	0	0
Production	01MAR04	01MAR04	01MAR04	01MAR04	0	0
Marketing	02FEB04	02FEB04	01MAR04	01MAR04	20	20

In fact, you can specify an alternate calendar for *all* the lag durations by using the ALAGCAL= or NLAGCAL= option in the SUCCESSOR statement. The next invocation of the CPM procedure illustrates this feature by specifying ALAGCAL=SEVENDAY in the SUCCESSOR statement. Thus, all the lag durations now follow the seven-day calendar instead of the five-day calendar, which is the default calendar for this project. Output 4.11.5 shows the resulting schedule. Now the project finishes on February 27, 2004. Output 4.11.6 displays a precedence Gantt chart of the project. Note how the nonstandard precedence constraints are displayed.

```
proc cpm data=widglag date='1dec03'd calendar=calendar
         interval=weekday collapse out=lagsched;
   activity task;
   succ     succ / lag = (lagdur) alagcal=sevenday;
   duration days;
run;

pattern1 c=green v=s;     /* duration of a non-critical activity   */
pattern2 c=green v=e;     /* slack time for a noncrit. activity    */
pattern3 c=red   v=s;     /* duration of a critical activity       */

title  h=1.5   'Non-Standard Relationships';
title2 h=1 'Precedence Gantt Chart';
```

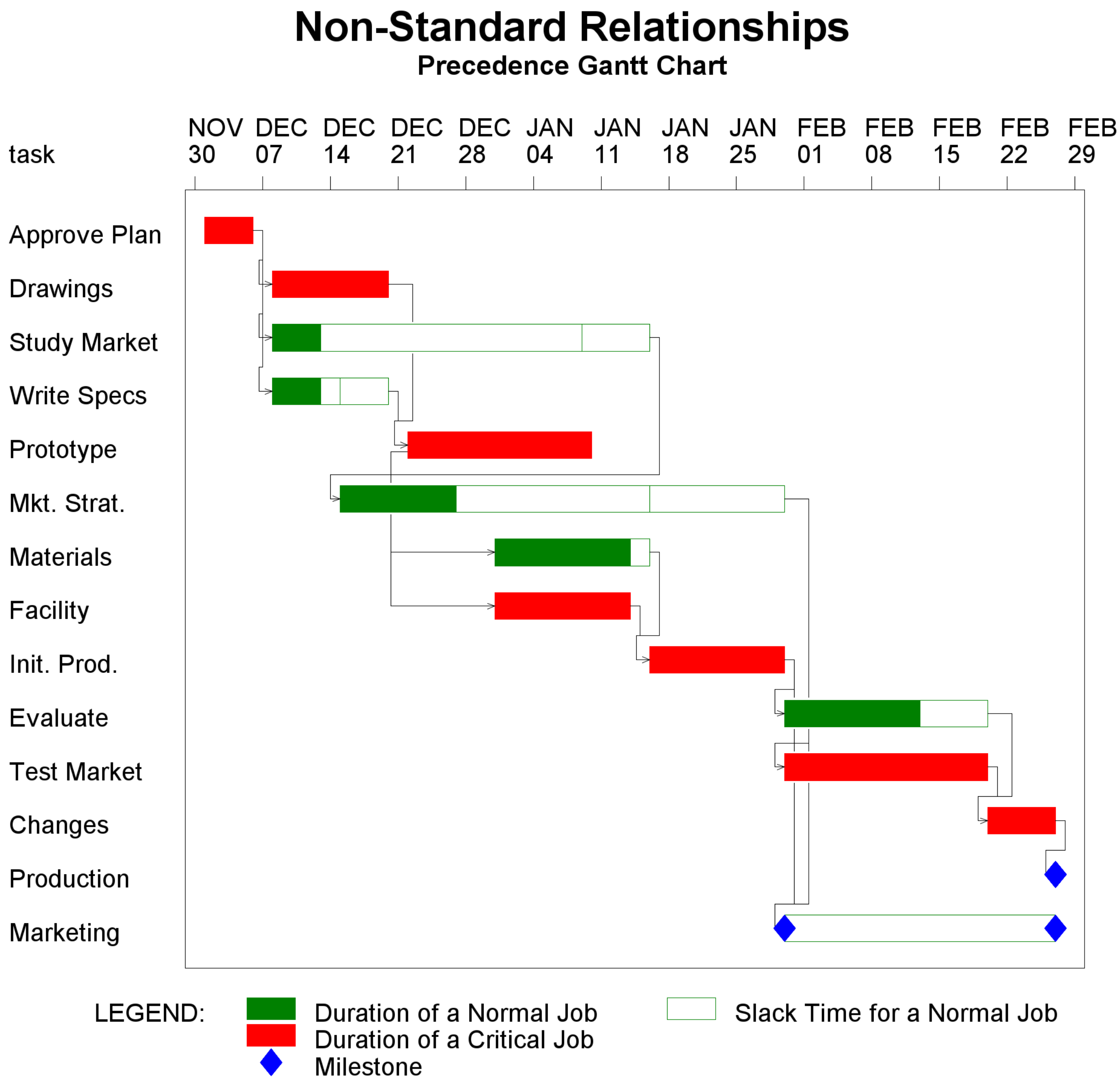
```
proc gantt graphics data=lagsched logic=widglag;
   chart / compress act=task succ=(succ) dur=days
           cprec=black cmile=blue
           caxis=black
           height=1.5 nojobnum
           dur=days increment=7 lag=(lagdur);
   id task;
   run;
```

Output 4.11.5 Project Schedule: LAG Calendar = SEVENDAY

Non-Standard Relationships
Lag Type and Duration: LAG Calendar = SEVENDAY

task	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Approve Plan	01DEC03	05DEC03	01DEC03	05DEC03	0	0
Drawings	08DEC03	19DEC03	08DEC03	19DEC03	0	0
Study Market	08DEC03	12DEC03	09JAN04	15JAN04	24	0
Write Specs	08DEC03	12DEC03	15DEC03	19DEC03	5	5
Prototype	22DEC03	09JAN04	22DEC03	09JAN04	0	0
Mkt. Strat.	15DEC03	26DEC03	16JAN04	29JAN04	24	24
Materials	31DEC03	13JAN04	02JAN04	15JAN04	2	2
Facility	31DEC03	13JAN04	31DEC03	13JAN04	0	0
Init. Prod.	16JAN04	29JAN04	16JAN04	29JAN04	0	0
Evaluate	30JAN04	12FEB04	06FEB04	19FEB04	5	5
Test Market	30JAN04	19FEB04	30JAN04	19FEB04	0	0
Changes	20FEB04	26FEB04	20FEB04	26FEB04	0	0
Production	27FEB04	27FEB04	27FEB04	27FEB04	0	0
Marketing	30JAN04	30JAN04	27FEB04	27FEB04	20	20

Output 4.11.6 Precedence Gantt Chart



Example 4.12: Activity Time Constraints

Often, in addition to a project start date or a project finish date, there may be other time constraints imposed selectively on the activities in the project. The ALIGNDATE and ALIGNTYPE statements enable you to add various types of time constraints on the activities. In this example, the data set WIDGET12 displayed in Output 4.12.1 contains two variables, adate and atype, which enable you to specify these restrictions. For example, the activity 'Drawings' has an 'feq' (Finish Equals) constraint, requiring it to finish on the 15th of December. The activity 'Test Market' has a *mandatory* start date imposed on it.

Output 4.12.1 Activity Data Set WIDGET12

Activity Time Constraints
Activity data set

Obs	task	days	succ1	succ2	succ3	adate	atype
1	Approve Plan	5	Drawings	Study Market	Write Specs	.	
2	Drawings	10	Prototype			15DEC03	feq
3	Study Market	5	Mkt. Strat.			.	
4	Write Specs	5	Prototype			15DEC03	sge
5	Prototype	15	Materials	Facility		.	
6	Mkt. Strat.	10	Test Market	Marketing		.	
7	Materials	10	Init. Prod.			.	
8	Facility	10	Init. Prod.			.	
9	Init. Prod.	10	Test Market	Marketing	Evaluate	.	
10	Evaluate	10	Changes			27FEB04	fle
11	Test Market	15	Changes			16FEB04	ms
12	Changes	5	Production			.	
13	Production	0				.	
14	Marketing	0				.	

The following statements are needed to schedule the project subject to these restrictions. The option XFERVARS in the PROC CPM statement causes CPM to transfer all variables that were used in the analysis to the Schedule data set. Output 4.12.2 shows the resulting schedule.

```
proc cpm data=widget12 date='1dec03'd
     xfervars interval=weekday;
   activity task;
   successor succ1 succ2 succ3;
   duration days;
   aligndate adate;
   aligntype atype;
   run;
```

```
title 'Activity Time Constraints';
title2 'Aligned Schedule';
proc print;
   id task;
   var adate atype e_: l_: t_float f_float;
   run;
```

Output 4.12.2 Aligned Schedule

Activity Time Constraints
Aligned Schedule

task	adate	atype	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Approve Plan	.		01DEC03	05DEC03	25NOV03	01DEC03	-4	-4
Drawings	15DEC03	feq	08DEC03	19DEC03	02DEC03	15DEC03	-4	-4
Study Market	.		08DEC03	12DEC03	26JAN04	30JAN04	35	0
Write Specs	15DEC03	sge	15DEC03	19DEC03	22DEC03	26DEC03	5	0
Prototype	.		22DEC03	09JAN04	29DEC03	16JAN04	5	0
Mkt. Strat.	.		15DEC03	26DEC03	02FEB04	13FEB04	35	30
Materials	.		12JAN04	23JAN04	19JAN04	30JAN04	5	0
Facility	.		12JAN04	23JAN04	19JAN04	30JAN04	5	0
Init. Prod.	.		26JAN04	06FEB04	02FEB04	13FEB04	5	0
Evaluate	27FEB04	fle	09FEB04	20FEB04	16FEB04	27FEB04	5	5
Test Market	16FEB04	ms	16FEB04	05MAR04	16FEB04	05MAR04	0	0
Changes	.		08MAR04	12MAR04	08MAR04	12MAR04	0	0
Production	.		15MAR04	15MAR04	15MAR04	15MAR04	0	0
Marketing	.		09FEB04	09FEB04	15MAR04	15MAR04	25	25

Note that the MS and MF constraints are *mandatory* and override any precedence constraints; thus, both the late start and early start times for the activity 'Test Market' coincide with February 16, 2004. However, the other types of constraints are not mandatory; they are superseded by any constraints imposed by the precedence relationships. In other words, neither the early start nor the late start schedule violate precedence constraints. Thus, even though the activity 'Drawings' is required to finish on the 15th of December (by the 'feq' constraint), the early start schedule causes it to finish on the 19th of December because of its predecessor's schedule. This type of inconsistency is indicated by the presence of negative floats for some of the activities alerting you to the fact that if some of these deadlines are to be met, these activities must start earlier than the early start schedule. Such activities are called *supercritical*.

Example 4.13: Progress Update and Target Schedules

This example shows the use of the ACTUAL and BASELINE statements to track and compare a project's progress with the original planned schedule. Consider the data in Example 4.1, for the network in AON format. Suppose that the project has started as scheduled on December 1, 2003, and that the current date is December 19, 2003. You may want to enter the actual dates for the activities that are already in progress or have been completed and use the CPM procedure to determine the schedule for activities that remain to be done. In addition to computing an updated schedule, you may want to check the progress of the project by comparing the current schedule with the planned schedule.

The BASELINE statement enables you to save a target schedule in the Schedule data set. In this example, suppose that you want to try to schedule the activities according to the project's early start schedule. As a first step, schedule the project with PROC CPM, and use the SET= option in the BASELINE statement to save the early start and finish times as the baseline start and finish times. The following program saves the baseline schedule (in the variables B_START and B_FINISH), and Output 4.13.1 displays the resulting output data set.

```
data holidays;
   format holiday holifin date7.;
   input holiday & date7. holifin & date7. holidur;
   datalines;
24dec03  26dec03  4
01jan04  .        .
;

* store early schedule as the baseline schedule;

proc cpm data=widget holidata=holidays
         out=widgbase date='1dec03'd;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   baseline / set=early;
   run;
```

Output 4.13.1 Target Schedule

Progress Update and Target Schedules
Set Baseline Schedule

Obs	task	succ1	succ2	succ3	days	E_START	E_FINISH
1	Approve Plan	Drawings	Study Market	Write Specs	5	01DEC03	05DEC03
2	Drawings	Prototype			10	06DEC03	15DEC03
3	Study Market	Mkt. Strat.			5	06DEC03	10DEC03
4	Write Specs	Prototype			5	06DEC03	10DEC03
5	Prototype	Materials	Facility		15	16DEC03	03JAN04
6	Mkt. Strat.	Test Market	Marketing		10	11DEC03	20DEC03
7	Materials	Init. Prod.			10	04JAN04	13JAN04
8	Facility	Init. Prod.			10	04JAN04	13JAN04
9	Init. Prod.	Test Market	Marketing	Evaluate	10	14JAN04	23JAN04
10	Evaluate	Changes			10	24JAN04	02FEB04
11	Test Market	Changes			15	24JAN04	07FEB04
12	Changes	Production			5	08FEB04	12FEB04
13	Production				0	13FEB04	13FEB04
14	Marketing				0	24JAN04	24JAN04

Obs	L_START	L_FINISH	T_FLOAT	F_FLOAT	B_START	B_FINISH
1	01DEC03	05DEC03	0	0	01DEC03	05DEC03
2	06DEC03	15DEC03	0	0	06DEC03	15DEC03
3	09JAN04	13JAN04	30	0	06DEC03	10DEC03
4	11DEC03	15DEC03	5	5	06DEC03	10DEC03
5	16DEC03	03JAN04	0	0	16DEC03	03JAN04
6	14JAN04	23JAN04	30	30	11DEC03	20DEC03
7	04JAN04	13JAN04	0	0	04JAN04	13JAN04
8	04JAN04	13JAN04	0	0	04JAN04	13JAN04
9	14JAN04	23JAN04	0	0	14JAN04	23JAN04
10	29JAN04	07FEB04	5	5	24JAN04	02FEB04
11	24JAN04	07FEB04	0	0	24JAN04	07FEB04
12	08FEB04	12FEB04	0	0	08FEB04	12FEB04
13	13FEB04	13FEB04	0	0	13FEB04	13FEB04
14	13FEB04	13FEB04	20	20	24JAN04	24JAN04

As the project progresses, you have to account for the actual progress of the project and schedule the unfinished activities accordingly. You can do so by specifying actual start or actual finish times (or both) for activities that have already finished or are in progress. Progress information can also be specified using percent complete or remaining duration values. Assume that current information has been incorporated into the ACTUAL data set, shown in Output 4.13.2. The variables sdate and fdate contain the actual start and finish times of the activities, and rdur specifies the number of days of work that are still remaining for the activity to be completed, and pctc specifies the percent of work that has been completed for that activity.

Output 4.13.2 Progress Data Set ACTUAL

Progress Update and Target Schedules
Progress Data

Obs	task	sdate	fdate	pctc	rdur
1	Approve Plan	01DEC2003	05DEC2003	.	.
2	Drawings	06DEC2003	16DEC2003	.	.
3	Study Market	05DEC2003	.	100	.
4	Write Specs	07DEC2003	12DEC2003	.	.
5	Prototype	.	.	.	.
6	Mkt. Strat.	10DEC2003	.	.	3
7	Materials	.	.	.	.
8	Facility	.	.	.	.
9	Init. Prod.	.	.	.	.
10	Evaluate	.	.	.	.
11	Test Market	.	.	.	.
12	Changes	.	.	.	.
13	Production	.	.	.	.
14	Marketing	.	.	.	.

The following statements invoke PROC CPM after merging the progress data with the Schedule data set. The NOAUTOUPDT option is specified so that only those activities that have explicit progress information are assumed to have started. The resulting Schedule data set contains the new variables A_START, A_FINISH, A_DUR, and STATUS; this data set is displayed in Output 4.13.3. The activity 'Mkt. Strat.', which has rdur='3' in Output 4.13.2, has an early finish time (December 21, 2003) that is three days after TIMENOW. The S_VAR and F_VAR variables show the amount of slippage in the start and finish times (predicted on the basis of the current schedule) as compared to the baseline schedule.

```
* merge the baseline information with progress update;
data widgact;
   merge  actual widgbase;
   run;

proc cpm data=widgact holidata=holidays
         out=widgnupd date='1dec03'd;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   baseline / compare=early;
   actual / a_start=sdate a_finish=fdate timenow='19dec03'd
            remdur=rdur pctcomp=pctc noautoupdt;
   run;
```

Output 4.13.3 Comparison of Schedules: NOAUTOUPDT

Progress Update and Target Schedules
Updated Schedule vs. Target Schedule: NOAUTOUPDT

Obs	task	succ1	succ2	succ3	days	STATUS	A_DUR	A_START	A_FINISH	E_START
1	Approve Plan	Drawings	Study Market	Write Specs	5	Completed	5	01DEC03	05DEC03	01DEC03
2	Drawings	Prototype			10	Completed	11	06DEC03	16DEC03	06DEC03
3	Study Market	Mkt. Strat.			5	Completed	5	05DEC03	09DEC03	05DEC03
4	Write Specs	Prototype			5	Completed	6	07DEC03	12DEC03	07DEC03
5	Prototype	Materials	Facility		15	Pending	.	.	.	19DEC03
6	Mkt. Strat.	Test Market	Marketing		10	In Progress	.	10DEC03	.	10DEC03
7	Materials	Init. Prod.			10	Pending	.	.	.	07JAN04
8	Facility	Init. Prod.			10	Pending	.	.	.	07JAN04
9	Init. Prod.	Test Market	Marketing	Evaluate	10	Pending	.	.	.	17JAN04
10	Evaluate	Changes			10	Pending	.	.	.	27JAN04
11	Test Market	Changes			15	Pending	.	.	.	27JAN04
12	Changes	Production			5	Pending	.	.	.	11FEB04
13	Production				0	Pending	.	.	.	16FEB04
14	Marketing				0	Pending	.	.	.	27JAN04

Obs	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT	B_START	B_FINISH	S_VAR	F_VAR
1	05DEC03	01DEC03	05DEC03	0	0	01DEC03	05DEC03	0	0
2	16DEC03	06DEC03	16DEC03	0	0	06DEC03	15DEC03	0	1
3	09DEC03	05DEC03	09DEC03	0	0	06DEC03	10DEC03	-1	-1
4	12DEC03	07DEC03	12DEC03	0	0	06DEC03	10DEC03	1	2
5	06JAN04	19DEC03	06JAN04	0	0	16DEC03	03JAN04	3	3
6	21DEC03	10DEC03	21DEC03	0	0	11DEC03	20DEC03	-1	1
7	16JAN04	07JAN04	16JAN04	0	0	04JAN04	13JAN04	3	3
8	16JAN04	07JAN04	16JAN04	0	0	04JAN04	13JAN04	3	3
9	26JAN04	17JAN04	26JAN04	0	0	14JAN04	23JAN04	3	3
10	05FEB04	01FEB04	10FEB04	5	5	24JAN04	02FEB04	3	3
11	10FEB04	27JAN04	10FEB04	0	0	24JAN04	07FEB04	3	3
12	15FEB04	11FEB04	15FEB04	0	0	08FEB04	12FEB04	3	3
13	16FEB04	16FEB04	16FEB04	0	0	13FEB04	13FEB04	3	3
14	27JAN04	16FEB04	16FEB04	20	20	24JAN04	24JAN04	3	3

In order for you to see the effect of the AUTOUPDT option, the same project information is used with the AUTOUPDT option in the ACTUAL statement. Output 4.13.4 displays the resulting schedule. With the AUTOUPDT option (which is, in fact, the default option), PROC CPM uses the progress information and the precedence information to automatically fill in the actual start and finish information for activities that should have finished or started before TIMENOW. The activity 'Prototype' has no progress information in WIDGACT, but it is assumed to have an actual start date of December 17, 2003. This option is useful when there are several activities that take place according to the plan and only a few occur out of sequence; then it is sufficient to enter progress information only for the activities that did not follow the plan. The SHOWFLOAT option, also used in this invocation of PROC CPM, enables activities that are completed or in progress to have float; in other words, the late start schedule for activities in progress is not fixed by the progress information. Thus, the activity 'Study Market' has L_START='08JAN04' instead of '05DEC03', as in the earlier invocation of PROC CPM (without the SHOWFLOAT option). The following invocation of PROC CPM produces Output 4.13.4:

```
proc cpm data=widgact holidata=holidays
         out=widgupdt date='1dec03'd;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   baseline / compare=early;
   actual / as=sdate af=fdate timenow='19dec03'd
            remdur=rdur pctcomp=pctc
            autoupdt showfloat;
   run;
```

Output 4.13.4 Comparison of Schedules: AUTOUPDT

Progress Update and Target Schedules
Updated Schedule vs. Target Schedule: AUTOUPDT

Obs	task	succ1	succ2	succ3	days	STATUS	A_DUR	A_START	A_FINISH	E_START
1	Approve Plan	Drawings	Study Market	Write Specs	5	Completed	5	01DEC03	05DEC03	01DEC03
2	Drawings	Prototype			10	Completed	11	06DEC03	16DEC03	06DEC03
3	Study Market	Mkt. Strat.			5	Completed	5	05DEC03	09DEC03	05DEC03
4	Write Specs	Prototype			5	Completed	6	07DEC03	12DEC03	07DEC03
5	Prototype	Materials	Facility		15	In Progress	.	17DEC03	.	17DEC03
6	Mkt. Strat.	Test Market	Marketing		10	In Progress	.	10DEC03	.	10DEC03
7	Materials	Init. Prod.			10	Pending	.	.	.	05JAN04
8	Facility	Init. Prod.			10	Pending	.	.	.	05JAN04
9	Init. Prod.	Test Market	Marketing	Evaluate	10	Pending	.	.	.	15JAN04
10	Evaluate	Changes			10	Pending	.	.	.	25JAN04
11	Test Market	Changes			15	Pending	.	.	.	25JAN04
12	Changes	Production			5	Pending	.	.	.	09FEB04
13	Production				0	Pending	.	.	.	14FEB04
14	Marketing				0	Pending	.	.	.	25JAN04

Obs	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT	B_START	B_FINISH	S_VAR	F_VAR
1	05DEC03	01DEC03	05DEC03	0	-1	01DEC03	05DEC03	0	0
2	16DEC03	06DEC03	16DEC03	0	0	06DEC03	15DEC03	0	1
3	09DEC03	08JAN04	12JAN04	30	0	06DEC03	10DEC03	-1	-1
4	12DEC03	11DEC03	16DEC03	4	4	06DEC03	10DEC03	1	2
5	04JAN04	17DEC03	04JAN04	0	0	16DEC03	03JAN04	1	1
6	21DEC03	13JAN04	24JAN04	30	30	11DEC03	20DEC03	-1	1
7	14JAN04	05JAN04	14JAN04	0	0	04JAN04	13JAN04	1	1
8	14JAN04	05JAN04	14JAN04	0	0	04JAN04	13JAN04	1	1
9	24JAN04	15JAN04	24JAN04	0	0	14JAN04	23JAN04	1	1
10	03FEB04	30JAN04	08FEB04	5	5	24JAN04	02FEB04	1	1
11	08FEB04	25JAN04	08FEB04	0	0	24JAN04	07FEB04	1	1
12	13FEB04	09FEB04	13FEB04	0	0	08FEB04	12FEB04	1	1
13	14FEB04	14FEB04	14FEB04	0	0	13FEB04	13FEB04	1	1
14	25JAN04	14FEB04	14FEB04	20	20	24JAN04	24JAN04	1	1

Example 4.14: Summarizing Resource Utilization

This example shows how you can use the RESOURCE statement in conjunction with the RESOURCEOUT= option to summarize resource utilization. The example assumes that Engineer is a resource category and the project network (in AOA format) along with resource requirements for each activity is in a SAS data set, as displayed in Output 4.14.1.

Output 4.14.1 Resource Utilization: WIDGRES

Summarizing Resource Utilization
Activity Data Set

Obs	task	days	tail	head	engineer
1	Approve Plan	5	1	2	2
2	Drawings	10	2	3	1
3	Study Market	5	2	4	1
4	Write Specs	5	2	3	2
5	Prototype	15	3	5	4
6	Mkt. Strat.	10	4	6	.
7	Materials	10	5	7	.
8	Facility	10	5	7	2
9	Init. Prod.	10	7	8	4
10	Evaluate	10	8	9	1
11	Test Market	15	6	9	.
12	Changes	5	9	10	2
13	Production	0	10	11	4
14	Marketing	0	6	12	.
15	Dummy	0	8	6	.

Output 4.14.2 Resource Utilization: HOLDATA

Summarizing Resource Utilization
Holidays Data Set HOLDATA

Obs	hol	name
1	25DEC03	Christmas
2	01JAN04	New Year

In the following program, PROC CPM is invoked with the RESOURCE statement identifying the resource for which usage information is required. The project is scheduled only on weekdays, and holiday information is included through the Holiday data set, HOLDATA, which identifies two holidays, one for Christmas and one for New Year's Day. Output 4.14.2 shows the Holiday data set.

The program saves the resource usage information in a data set named ROUT, which is displayed in Output 4.14.3. Two variables, Eengineer and Lengineer, denote the usage of the resource engineer corresponding to the early and late start schedules, respectively. Note the naming convention for the variables in the resource usage data set: A prefix (E for Early and L for Late) is followed by the name of the resource variable, engineer. Note also that the data set contains only observations corresponding to weekdays; by default, the _TIME_ variable in the resource usage output data set increases by one unit *interval* of the default calendar for every observation. Further, the MAXDATE= option is used in the RESOURCE statement to get resource usage information only for the month of December.

```
proc cpm date='1dec03'd interval=weekday
         resourceout=rout data=widgres
         holidata=holdata;
   id task;
   tailnode tail;
   duration days;
   headnode head;
   resource engineer / maxdate='31dec03'd;
   holiday  hol;
   run;
```

Output 4.14.3 Resource Utilization: Resource Usage Data Set

Summarizing Resource Utilization
Resource Usage

Obs	_TIME_	Eengineer	Lengineer
1	01DEC03	2	2
2	02DEC03	2	2
3	03DEC03	2	2
4	04DEC03	2	2
5	05DEC03	2	2
6	08DEC03	4	1
7	09DEC03	4	1
8	10DEC03	4	1
9	11DEC03	4	1
10	12DEC03	4	1
11	15DEC03	1	3
12	16DEC03	1	3
13	17DEC03	1	3
14	18DEC03	1	3
15	19DEC03	1	3
16	22DEC03	4	4
17	23DEC03	4	4
18	24DEC03	4	4
19	26DEC03	4	4
20	29DEC03	4	4
21	30DEC03	4	4
22	31DEC03	4	4

This data set can be used as input for any type of resource utilization report. In this example, the resource usage for the month of December is presented in two ways: on a calendar and in a chart. The following program prints the calendar and bar chart:

```
/* format the Engineer variables */
proc format;
   picture efmt other='9 ESS Eng.';
   picture lfmt other='9 LSS Eng.';

proc calendar legend weekdays
     data=rout holidata=holdata;
   id _time_;
   var  eengineer lengineer;
   format eengineer efmt. lengineer lfmt.;
   holiday hol;
   holiname name;

proc chart data=rout;
   hbar _time_/sumvar=eengineer discrete;
   hbar _time_/sumvar=lengineer discrete;
   run;
```

Output 4.14.4 Calendar Showing Resource Usage

Summarizing Resource Utilization
Resource Usage

```
-------------------------------------------------------------
|                                                           |
|                      December  2003                        |
|                                                           |
|-----------------------------------------------------------|
|  Monday   |  Tuesday  | Wednesday | Thursday  |  Friday   |
|-----------+-----------+-----------+-----------+-----------|
|     1     |     2     |     3     |     4     |     5     |
|           |           |           |           |           |
| 2 ESS Eng | 2 ESS Eng | 2 ESS Eng | 2 ESS Eng | 2 ESS Eng |
| 2 LSS Eng | 2 LSS Eng | 2 LSS Eng | 2 LSS Eng | 2 LSS Eng |
|-----------+-----------+-----------+-----------+-----------|
|     8     |     9     |    10     |    11     |    12     |
|           |           |           |           |           |
| 4 ESS Eng | 4 ESS Eng | 4 ESS Eng | 4 ESS Eng | 4 ESS Eng |
| 1 LSS Eng | 1 LSS Eng | 1 LSS Eng | 1 LSS Eng | 1 LSS Eng |
|-----------+-----------+-----------+-----------+-----------|
|    15     |    16     |    17     |    18     |    19     |
|           |           |           |           |           |
| 1 ESS Eng | 1 ESS Eng | 1 ESS Eng | 1 ESS Eng | 1 ESS Eng |
| 3 LSS Eng | 3 LSS Eng | 3 LSS Eng | 3 LSS Eng | 3 LSS Eng |
|-----------+-----------+-----------+-----------+-----------|
|    22     |    23     |    24     |    25     |    26     |
|           |           |           |*Christmas*|           |
| 4 ESS Eng | 4 ESS Eng | 4 ESS Eng |           | 4 ESS Eng |
| 4 LSS Eng | 4 LSS Eng | 4 LSS Eng |           | 4 LSS Eng |
|-----------+-----------+-----------+-----------+-----------|
|    29     |    30     |    31     |           |           |
|           |           |           |           |           |
| 4 ESS Eng | 4 ESS Eng | 4 ESS Eng |           |           |
| 4 LSS Eng | 4 LSS Eng | 4 LSS Eng |           |           |
-------------------------------------------------------------
```

```
-------------------------
|        Legend         |
|                       |
| ESS Usage of engineer |
| LSS Usage of engineer |
-------------------------
```

Output 4.14.5 Bar Chart for Early Start Usage

Summarizing Resource Utilization
Resource Usage

```
 Period Identifier                                    ESS Usage of  en
                                                                   Sum
           |
01DEC03    |********************                              2.000000
02DEC03    |********************                              2.000000
03DEC03    |********************                              2.000000
04DEC03    |********************                              2.000000
05DEC03    |********************                              2.000000
08DEC03    |****************************************          4.000000
09DEC03    |****************************************          4.000000
10DEC03    |****************************************          4.000000
11DEC03    |****************************************          4.000000
12DEC03    |****************************************          4.000000
15DEC03    |**********                                        1.000000
16DEC03    |**********                                        1.000000
17DEC03    |**********                                        1.000000
18DEC03    |**********                                        1.000000
19DEC03    |**********                                        1.000000
22DEC03    |****************************************          4.000000
23DEC03    |****************************************          4.000000
24DEC03    |****************************************          4.000000
26DEC03    |****************************************          4.000000
29DEC03    |****************************************          4.000000
30DEC03    |****************************************          4.000000
31DEC03    |****************************************          4.000000
           |
           ----------+---------+---------+---------+
                     1         2         3         4

                     ESS Usage of  engineer
```

Output 4.14.6 Bar Chart for Late Start Usage

```
                    Summarizing Resource Utilization
                             Resource Usage

 Period Identifier                                   LSS Usage of  en
                                                                  Sum
           |
 01DEC03   |********************                            2.000000
 02DEC03   |********************                            2.000000
 03DEC03   |********************                            2.000000
 04DEC03   |********************                            2.000000
 05DEC03   |********************                            2.000000
 08DEC03   |**********                                      1.000000
 09DEC03   |**********                                      1.000000
 10DEC03   |**********                                      1.000000
 11DEC03   |**********                                      1.000000
 12DEC03   |**********                                      1.000000
 15DEC03   |******************************                  3.000000
 16DEC03   |******************************                  3.000000
 17DEC03   |******************************                  3.000000
 18DEC03   |******************************                  3.000000
 19DEC03   |******************************                  3.000000
 22DEC03   |****************************************        4.000000
 23DEC03   |****************************************        4.000000
 24DEC03   |****************************************        4.000000
 26DEC03   |****************************************        4.000000
 29DEC03   |****************************************        4.000000
 30DEC03   |****************************************        4.000000
 31DEC03   |****************************************        4.000000
           |
           ----------+---------+---------+---------+
                     1         2         3         4

                     LSS Usage of  engineer
```

Charts such as those shown in Output 4.14.4 through Output 4.14.6 can be used to compare different schedules with respect to resource usage.

Example 4.15: Resource Allocation

In the previous example, a summary of the resource utilization is obtained. Suppose that you want to schedule the project subject to constraints on the availability of ENGINEERS. The activity data, as in Example 4.14, are assumed to be in a data set named WIDGRES. The resource variable, engineer, specifies the number of engineers needed per day for each activity in the project. In addition to the resource engineer, a consumable resource engcost is computed at a daily rate of 200 for each unit of resource engineer used per day. The following DATA step uses the Activity data set from Example 4.14 to create a new Activity data set that includes the resource engcost.

```
data widgres;
set widgres;
if engineer ^= . then engcost = engineer * 200;
run;
```

Now suppose that the availability of the resource engineer and the total outlay for engcost is saved in a data set named WIDGRIN, displayed in Output 4.15.1.

Output 4.15.1 Resource Availability Data Set

Resource Allocation
Data Set WIDGRIN

Obs	per	otype	engineer	engcost
1	.	restype	1	2
2	.	suplevel	1	.
3	01DEC03	reslevel	3	40000
4	26DEC03	reslevel	4	.

In the data set WIDGRIN, the first observation indicates that engineer is a replenishable resource, while engcost is a consumable resource. The second observation indicates that an extra engineer is available, if necessary. The remaining observations indicate the availability profile starting from December 1, 2003. PROC CPM is then used to schedule the project to start on December 1, 2003, subject to the availability as specified.

```
proc cpm date='01dec03'd interval=weekday
         data=widgres holidata=holdata resin=widgrin
         out=widgschd resout=widgrout;
   tailnode tail;
   duration days;
   headnode head;
   holiday hol;
   resource engineer engcost / period=per obstype=otype
                               schedrule=shortdur
                               delayanalysis;
   id task;
   run;
```

Output 4.15.2 Resource Constrained Schedule: Rule = SHORTDUR

Resource Allocation
Resource Constrained Schedule: Rule = SHORTDUR

Obs	tail	head	days	task	engineer	engcost	S_START	S_FINISH	E_START	E_FINISH
1	1	2	5	Approve Plan	2	400	01DEC03	05DEC03	01DEC03	05DEC03
2	2	3	10	Drawings	1	200	15DEC03	29DEC03	08DEC03	19DEC03
3	2	4	5	Study Market	1	200	08DEC03	12DEC03	08DEC03	12DEC03
4	2	3	5	Write Specs	2	400	08DEC03	12DEC03	08DEC03	12DEC03
5	3	5	15	Prototype	4	800	30DEC03	20JAN04	22DEC03	13JAN04
6	4	6	10	Mkt. Strat.	.	.	15DEC03	29DEC03	15DEC03	29DEC03
7	5	7	10	Materials	.	.	21JAN04	03FEB04	14JAN04	27JAN04
8	5	7	10	Facility	2	400	21JAN04	03FEB04	14JAN04	27JAN04
9	7	8	10	Init. Prod.	4	800	04FEB04	17FEB04	28JAN04	10FEB04
10	8	9	10	Evaluate	1	200	18FEB04	02MAR04	11FEB04	24FEB04
11	6	9	15	Test Market	.	.	18FEB04	09MAR04	11FEB04	02MAR04
12	9	10	5	Changes	2	400	10MAR04	16MAR04	03MAR04	09MAR04
13	10	11	0	Production	4	800	17MAR04	17MAR04	10MAR04	10MAR04
14	6	12	0	Marketing	.	.	18FEB04	18FEB04	11FEB04	11FEB04
15	8	6	0	Dummy	.	.	18FEB04	18FEB04	11FEB04	11FEB04

Obs	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	01DEC03	05DEC03	0		
2	08DEC03	19DEC03	5	engineer	
3	21JAN04	27JAN04	0		
4	15DEC03	19DEC03	0		
5	22DEC03	13JAN04	0		
6	28JAN04	10FEB04	0		
7	14JAN04	27JAN04	0		
8	14JAN04	27JAN04	0		
9	28JAN04	10FEB04	0		
10	18FEB04	02MAR04	0		
11	11FEB04	02MAR04	0		
12	03MAR04	09MAR04	0		
13	10MAR04	10MAR04	0		
14	10MAR04	10MAR04	0		
15	11FEB04	11FEB04	0		

In the first invocation of PROC CPM, the scheduling rule used for ordering the activities to be scheduled at a given time is specified to be SHORTDUR. The data set WIDGSCHD, displayed in Output 4.15.2, contains the resource constrained start and finish times in the variables S_START and S_FINISH. On December 8, three activities can be scheduled, all of which require the resource engineer. Using the scheduling rule specified, PROC CPM schedules the activities with the shortest durations first; thus, the activity 'Drawings' is delayed by five working days, until December 15, 2003.

The DELAYANALYSIS option in the RESOURCE statement helps analyze the cause of the delay by adding three new variables to the Schedule data set, R_DELAY, DELAY_R, and SUPPL_R. In this example, the R_DELAY and DELAY_R variables indicate that there is a delay of five days in the activity 'Drawings' due to the resource engineer. Such information helps to pinpoint the source of resource insufficiency, if any.

Other activities that follow 'Drawings' also have S_START>E_START, but the slippage in these activities is not caused by resource insufficiency, it is due to their predecessors being delayed. The entire project is delayed by five working days due to resource constraints (the maximum value of S_FINISH is 17MAR04, while the maximum value of E_FINISH is 10MAR04).

In this invocation, the DELAY= option is not specified; therefore, the supplementary level of resource is not used, since the primary levels of resources are found to be sufficient to schedule the project by delaying some of the activities.

The data set WIDGROUT, displayed in Output 4.15.3, contains variables Rengineer and Aengineer in addition to the variables Eengineer and Lengineer. The variable Rengineer denotes the usage of the resource engineer corresponding to the resource-constrained schedule, and Aengineer denotes the remaining level of the resource after resource allocation. For the consumable resource engcost, the variables Eengcost, Lengcost, and Rengcost indicate the rate of usage per unit *routinterval* (which defaults to INTERVAL=WEEKDAY, in this case) at the start of the time interval specified in the variable _TIME_. The variable Aengcost denotes the amount of money available at the beginning of the time specified in the _TIME_ variable.

Output 4.15.3 Resource Usage: Rule = SHORTDUR

Resource Allocation
Usage Profiles for Constrained Schedule: Rule = SHORTDUR

Obs	_TIME_	Eengineer	Lengineer	Rengineer	Aengineer	Eengcost	Lengcost	Rengcost	Aengcost
1	01DEC03	2	2	2	1	400	400	400	40000
2	02DEC03	2	2	2	1	400	400	400	39600
3	03DEC03	2	2	2	1	400	400	400	39200
4	04DEC03	2	2	2	1	400	400	400	38800
5	05DEC03	2	2	2	1	400	400	400	38400
6	08DEC03	4	1	3	0	800	200	600	38000
7	09DEC03	4	1	3	0	800	200	600	37400
8	10DEC03	4	1	3	0	800	200	600	36800
9	11DEC03	4	1	3	0	800	200	600	36200
10	12DEC03	4	1	3	0	800	200	600	35600
11	15DEC03	1	3	1	2	200	600	200	35000
12	16DEC03	1	3	1	2	200	600	200	34800
13	17DEC03	1	3	1	2	200	600	200	34600
14	18DEC03	1	3	1	2	200	600	200	34400
15	19DEC03	1	3	1	2	200	600	200	34200
16	22DEC03	4	4	1	2	800	800	200	34000
17	23DEC03	4	4	1	2	800	800	200	33800
18	24DEC03	4	4	1	2	800	800	200	33600
19	26DEC03	4	4	1	3	800	800	200	33400
20	29DEC03	4	4	1	3	800	800	200	33200
21	30DEC03	4	4	4	0	800	800	800	33000
22	31DEC03	4	4	4	0	800	800	800	32200
23	02JAN04	4	4	4	0	800	800	800	31400
24	05JAN04	4	4	4	0	800	800	800	30600
25	06JAN04	4	4	4	0	800	800	800	29800
26	07JAN04	4	4	4	0	800	800	800	29000
27	08JAN04	4	4	4	0	800	800	800	28200
28	09JAN04	4	4	4	0	800	800	800	27400
29	12JAN04	4	4	4	0	800	800	800	26600
30	13JAN04	4	4	4	0	800	800	800	25800
31	14JAN04	2	2	4	0	400	400	800	25000
32	15JAN04	2	2	4	0	400	400	800	24200
33	16JAN04	2	2	4	0	400	400	800	23400
34	19JAN04	2	2	4	0	400	400	800	22600
35	20JAN04	2	2	4	0	400	400	800	21800
36	21JAN04	2	3	2	2	400	600	400	21000
37	22JAN04	2	3	2	2	400	600	400	20600
38	23JAN04	2	3	2	2	400	600	400	20200
39	26JAN04	2	3	2	2	400	600	400	19800
40	27JAN04	2	3	2	2	400	600	400	19400

Output 4.15.3 *continued*

Resource Allocation
Usage Profiles for Constrained Schedule: Rule = SHORTDUR

Obs	_TIME_	Eengineer	Lengineer	Rengineer	Aengineer	Engcost	Lengcost	Rengcost	Aengcost
41	28JAN04	4	4	2	2	800	800	400	19000
42	29JAN04	4	4	2	2	800	800	400	18600
43	30JAN04	4	4	2	2	800	800	400	18200
44	02FEB04	4	4	2	2	800	800	400	17800
45	03FEB04	4	4	2	2	800	800	400	17400
46	04FEB04	4	4	4	0	800	800	800	17000
47	05FEB04	4	4	4	0	800	800	800	16200
48	06FEB04	4	4	4	0	800	800	800	15400
49	09FEB04	4	4	4	0	800	800	800	14600
50	10FEB04	4	4	4	0	800	800	800	13800
51	11FEB04	1	0	4	0	200	0	800	13000
52	12FEB04	1	0	4	0	200	0	800	12200
53	13FEB04	1	0	4	0	200	0	800	11400
54	16FEB04	1	0	4	0	200	0	800	10600
55	17FEB04	1	0	4	0	200	0	800	9800
56	18FEB04	1	1	1	3	200	200	200	9000
57	19FEB04	1	1	1	3	200	200	200	8800
58	20FEB04	1	1	1	3	200	200	200	8600
59	23FEB04	1	1	1	3	200	200	200	8400
60	24FEB04	1	1	1	3	200	200	200	8200
61	25FEB04	0	1	1	3	0	200	200	8000
62	26FEB04	0	1	1	3	0	200	200	7800
63	27FEB04	0	1	1	3	0	200	200	7600
64	01MAR04	0	1	1	3	0	200	200	7400
65	02MAR04	0	1	1	3	0	200	200	7200
66	03MAR04	2	2	0	4	400	400	0	7000
67	04MAR04	2	2	0	4	400	400	0	7000
68	05MAR04	2	2	0	4	400	400	0	7000
69	08MAR04	2	2	0	4	400	400	0	7000
70	09MAR04	2	2	0	4	400	400	0	7000
71	10MAR04	0	0	2	2	0	0	400	7000
72	11MAR04	0	0	2	2	0	0	400	6600
73	12MAR04	0	0	2	2	0	0	400	6200
74	15MAR04	0	0	2	2	0	0	400	5800
75	16MAR04	0	0	2	2	0	0	400	5400
76	17MAR04	0	0	0	4	0	0	0	5000

The second invocation of PROC CPM uses a different scheduling rule (LST, which is the default scheduling rule). Ties are broken using the L_START times for the activities. In this example, this rule results in a shorter project schedule. The schedule and the resource usage data sets are displayed in Output 4.15.4 and Output 4.15.5, respectively. Once again the variables DELAY_R and R_DELAY indicate that the resource engineer caused the activity ‘Study Market’ (‘Prototype’) to be delayed by five days (three days). However, the entire project is delayed only by three working days because the activity ‘Study Market’ is not a critical activity, and delaying it by five days did not affect the project completion time. Even with the resource delay of 5 days, this activity is scheduled earlier (S_START=15DEC03) than its latest start time (L_START=21JAN04).

```
proc cpm date='01dec03'd
         interval=weekday
         data=widgres
         resin=widgrin
         holidata=holdata
         out=widgsch2
         resout=widgrou2;
   tailnode tail;
   duration days;
   headnode head;
   holiday hol;
   resource engineer engcost / period=per
                               obstype=otype
                               schedrule=lst
                               delayanalysis;
   id task;
   run;
```

Output 4.15.4 Resource Constrained Schedule: Rule = LST

Resource Allocation
Resource Constrained Schedule: Rule = LST

Obs	tail	head	days	task	engineer	engcost	S_START	S_FINISH	E_START	E_FINISH
1	1	2	5	Approve Plan	2	400	01DEC03	05DEC03	01DEC03	05DEC03
2	2	3	10	Drawings	1	200	08DEC03	19DEC03	08DEC03	19DEC03
3	2	4	5	Study Market	1	200	15DEC03	19DEC03	08DEC03	12DEC03
4	2	3	5	Write Specs	2	400	08DEC03	12DEC03	08DEC03	12DEC03
5	3	5	15	Prototype	4	800	26DEC03	16JAN04	22DEC03	13JAN04
6	4	6	10	Mkt. Strat.	.	.	22DEC03	06JAN04	15DEC03	29DEC03
7	5	7	10	Materials	.	.	19JAN04	30JAN04	14JAN04	27JAN04
8	5	7	10	Facility	2	400	19JAN04	30JAN04	14JAN04	27JAN04
9	7	8	10	Init. Prod.	4	800	02FEB04	13FEB04	28JAN04	10FEB04
10	8	9	10	Evaluate	1	200	16FEB04	27FEB04	11FEB04	24FEB04
11	6	9	15	Test Market	.	.	16FEB04	05MAR04	11FEB04	02MAR04
12	9	10	5	Changes	2	400	08MAR04	12MAR04	03MAR04	09MAR04
13	10	11	0	Production	4	800	15MAR04	15MAR04	10MAR04	10MAR04
14	6	12	0	Marketing	.	.	16FEB04	16FEB04	11FEB04	11FEB04
15	8	6	0	Dummy	.	.	16FEB04	16FEB04	11FEB04	11FEB04

Obs	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	01DEC03	05DEC03	0		
2	08DEC03	19DEC03	0		
3	21JAN04	27JAN04	5	engineer	
4	15DEC03	19DEC03	0		
5	22DEC03	13JAN04	3	engineer	
6	28JAN04	10FEB04	0		
7	14JAN04	27JAN04	0		
8	14JAN04	27JAN04	0		
9	28JAN04	10FEB04	0		
10	18FEB04	02MAR04	0		
11	11FEB04	02MAR04	0		
12	03MAR04	09MAR04	0		
13	10MAR04	10MAR04	0		
14	10MAR04	10MAR04	0		
15	11FEB04	11FEB04	0		

Output 4.15.5 Resource Usage: Rule = LST

Resource Allocation
Usage Profiles for Constrained Schedule: Rule = LST

Obs	_TIME_	Eengineer	Lengineer	Rengineer	Aengineer	Engcost	Lengcost	Rengcost	Aengcost
1	01DEC03	2	2	2	1	400	400	400	40000
2	02DEC03	2	2	2	1	400	400	400	39600
3	03DEC03	2	2	2	1	400	400	400	39200
4	04DEC03	2	2	2	1	400	400	400	38800
5	05DEC03	2	2	2	1	400	400	400	38400
6	08DEC03	4	1	3	0	800	200	600	38000
7	09DEC03	4	1	3	0	800	200	600	37400
8	10DEC03	4	1	3	0	800	200	600	36800
9	11DEC03	4	1	3	0	800	200	600	36200
10	12DEC03	4	1	3	0	800	200	600	35600
11	15DEC03	1	3	2	1	200	600	400	35000
12	16DEC03	1	3	2	1	200	600	400	34600
13	17DEC03	1	3	2	1	200	600	400	34200
14	18DEC03	1	3	2	1	200	600	400	33800
15	19DEC03	1	3	2	1	200	600	400	33400
16	22DEC03	4	4	0	3	800	800	0	33000
17	23DEC03	4	4	0	3	800	800	0	33000
18	24DEC03	4	4	0	3	800	800	0	33000
19	26DEC03	4	4	4	0	800	800	800	33000
20	29DEC03	4	4	4	0	800	800	800	32200
21	30DEC03	4	4	4	0	800	800	800	31400
22	31DEC03	4	4	4	0	800	800	800	30600
23	02JAN04	4	4	4	0	800	800	800	29800
24	05JAN04	4	4	4	0	800	800	800	29000
25	06JAN04	4	4	4	0	800	800	800	28200
26	07JAN04	4	4	4	0	800	800	800	27400
27	08JAN04	4	4	4	0	800	800	800	26600
28	09JAN04	4	4	4	0	800	800	800	25800
29	12JAN04	4	4	4	0	800	800	800	25000
30	13JAN04	4	4	4	0	800	800	800	24200
31	14JAN04	2	2	4	0	400	400	800	23400
32	15JAN04	2	2	4	0	400	400	800	22600
33	16JAN04	2	2	4	0	400	400	800	21800
34	19JAN04	2	2	2	2	400	400	400	21000
35	20JAN04	2	2	2	2	400	400	400	20600
36	21JAN04	2	3	2	2	400	600	400	20200
37	22JAN04	2	3	2	2	400	600	400	19800
38	23JAN04	2	3	2	2	400	600	400	19400
39	26JAN04	2	3	2	2	400	600	400	19000
40	27JAN04	2	3	2	2	400	600	400	18600

Output 4.15.5 *continued*

Resource Allocation
Usage Profiles for Constrained Schedule: Rule = LST

Obs	_TIME_	Engineer	Leng ineer	Reng ineer	Aeng ineer	Engcost	Lengcost	Rengcost	Aengcost
41	28JAN04	4	4	2	2	800	800	400	18200
42	29JAN04	4	4	2	2	800	800	400	17800
43	30JAN04	4	4	2	2	800	800	400	17400
44	02FEB04	4	4	4	0	800	800	800	17000
45	03FEB04	4	4	4	0	800	800	800	16200
46	04FEB04	4	4	4	0	800	800	800	15400
47	05FEB04	4	4	4	0	800	800	800	14600
48	06FEB04	4	4	4	0	800	800	800	13800
49	09FEB04	4	4	4	0	800	800	800	13000
50	10FEB04	4	4	4	0	800	800	800	12200
51	11FEB04	1	0	4	0	200	0	800	11400
52	12FEB04	1	0	4	0	200	0	800	10600
53	13FEB04	1	0	4	0	200	0	800	9800
54	16FEB04	1	0	1	3	200	0	200	9000
55	17FEB04	1	0	1	3	200	0	200	8800
56	18FEB04	1	1	1	3	200	200	200	8600
57	19FEB04	1	1	1	3	200	200	200	8400
58	20FEB04	1	1	1	3	200	200	200	8200
59	23FEB04	1	1	1	3	200	200	200	8000
60	24FEB04	1	1	1	3	200	200	200	7800
61	25FEB04	0	1	1	3	0	200	200	7600
62	26FEB04	0	1	1	3	0	200	200	7400
63	27FEB04	0	1	1	3	0	200	200	7200
64	01MAR04	0	1	0	4	0	200	0	7000
65	02MAR04	0	1	0	4	0	200	0	7000
66	03MAR04	2	2	0	4	400	400	0	7000
67	04MAR04	2	2	0	4	400	400	0	7000
68	05MAR04	2	2	0	4	400	400	0	7000
69	08MAR04	2	2	2	2	400	400	400	7000
70	09MAR04	2	2	2	2	400	400	400	6600
71	10MAR04	0	0	2	2	0	0	400	6200
72	11MAR04	0	0	2	2	0	0	400	5800
73	12MAR04	0	0	2	2	0	0	400	5400
74	15MAR04	0	0	0	4	0	0	0	5000

Example 4.16: Using Supplementary Resources

In this example, the same project as in Example 4.15 is scheduled with a specification of DELAY=0. This indicates to PROC CPM that a supplementary level of resources is to be used if an activity cannot be scheduled to start on or before its latest start time (as computed in the unconstrained case). The schedule data and resource usage data are saved in the data sets WIDGO16 and WIDGRO16, respectively. They are displayed in Output 4.16.1 and Output 4.16.2, respectively.

```
title 'Using Supplementary Resources';
proc cpm date='01dec03'd interval=weekday
         data=widgres holidata=holdata resin=widgrin
         out=widgo16 resout=widgro16;
   tailnode tail;
   duration days;
   headnode head;
   holiday hol;
   resource engineer engcost / period=per obstype=otype
                               cumusage
                               delay=0
                               delayanalysis
                               routnobreak;
   id task;
   run;
```

To analyze the results of the resource constrained scheduling, you must examine both output data sets, WIDGRO16 and WIDGO16. The negative values for Aengineer in observation numbers 22 through 25 of the Usage data set WIDGRO16 indicate the amount of supplementary resource that is needed on December 22, 23, 24, and 25, to complete the project without delaying any activity beyond its latest start time. Examination of the SUPPL_R variable in the Schedule data set WIDGO16 indicates that the activity, ‘Prototype’, is scheduled to start on December 22 by using a supplementary level of the resource engineer.

The supplementary level is used only if the activity would otherwise get delayed beyond L_START + DELAY. Thus, the activity ‘Study Market’ is delayed by five days (S_START = ‘15DEC03’) and scheduled later than its early start time (E_START = ‘08DEC03’), even though a supplementary level of the resource could have been used to start the activity earlier, because the activity’s L_START time is equal to ‘21JAN04’ and DELAY = 0.

Further, note the use of the option CUMUSAGE in the RESOURCE statement, requesting that *cumulative* resource usage be saved in the Usage data set for consumable resources. Thus, for the consumable resource engcost, the procedure saves the *cumulative* resource usage in the variables Eengcost, Lengcost, and Rengcost, respectively. For instance, Eengcost in a given observation specifies the cumulative value of engcost for the early start schedule through the end of the previous day.

Output 4.16.1 Resource-Constrained Schedule: Supplementary Resource

Using Supplementary Resources
Resource Constrained Schedule

Obs	tail	head	days	task	engineer	engcost	S_START	S_FINISH	E_START	E_FINISH
1	1	2	5	Approve Plan	2	400	01DEC03	05DEC03	01DEC03	05DEC03
2	2	3	10	Drawings	1	200	08DEC03	19DEC03	08DEC03	19DEC03
3	2	4	5	Study Market	1	200	15DEC03	19DEC03	08DEC03	12DEC03
4	2	3	5	Write Specs	2	400	08DEC03	12DEC03	08DEC03	12DEC03
5	3	5	15	Prototype	4	800	22DEC03	13JAN04	22DEC03	13JAN04
6	4	6	10	Mkt. Strat.	.	.	22DEC03	06JAN04	15DEC03	29DEC03
7	5	7	10	Materials	.	.	14JAN04	27JAN04	14JAN04	27JAN04
8	5	7	10	Facility	2	400	14JAN04	27JAN04	14JAN04	27JAN04
9	7	8	10	Init. Prod.	4	800	28JAN04	10FEB04	28JAN04	10FEB04
10	8	9	10	Evaluate	1	200	11FEB04	24FEB04	11FEB04	24FEB04
11	6	9	15	Test Market	.	.	11FEB04	02MAR04	11FEB04	02MAR04
12	9	10	5	Changes	2	400	03MAR04	09MAR04	03MAR04	09MAR04
13	10	11	0	Production	4	800	10MAR04	10MAR04	10MAR04	10MAR04
14	6	12	0	Marketing	.	.	11FEB04	11FEB04	11FEB04	11FEB04
15	8	6	0	Dummy	.	.	11FEB04	11FEB04	11FEB04	11FEB04

Obs	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	01DEC03	05DEC03	0		
2	08DEC03	19DEC03	0		
3	21JAN04	27JAN04	5	engineer	
4	15DEC03	19DEC03	0		
5	22DEC03	13JAN04	0		engineer
6	28JAN04	10FEB04	0		
7	14JAN04	27JAN04	0		
8	14JAN04	27JAN04	0		
9	28JAN04	10FEB04	0		
10	18FEB04	02MAR04	0		
11	11FEB04	02MAR04	0		
12	03MAR04	09MAR04	0		
13	10MAR04	10MAR04	0		
14	10MAR04	10MAR04	0		
15	11FEB04	11FEB04	0		

This example also illustrates the use of the ROUTNOBREAK option to produce a resource usage output data set that does not have any breaks for holidays. Thus, the output data set WIDGRO16 has observations corresponding to holidays and weekends, unlike the corresponding resource output data sets in Example 4.15. Note that for consumable resources with cumulative usage there is no accumulation of the resource on holidays; thus, the cumulative value of engcost at the beginning of the 7th and 8th of December equals the value for the beginning of the 6th of December. For the resource engineer, however, the resource is assumed to be tied to the activity in progress even across holidays or weekends that are spanned by the activity's duration. For example, both activities 'Drawings' and 'Write Specs' start on December 8, 2003, requiring one and two engineers, respectively. The 'Write Specs' activity finishes on the 12th, freeing up two engineers, whereas 'Drawings' finishes only on the 19th of December. Thus, the data set WIDGRO16 has Rengineer equal to '3' from 8DEC03 to 12DEC03 and then equal to '1' on the 13th and 14th of December. Another engineer is required by the activity 'Study Market' from December 15, 2003; thus, the total usage from 15DEC03 to 19DEC03 is '2'.

Output 4.16.2 Resource Usage: Supplementary Resources

Using Supplementary Resources
Usage Profiles for Constrained Schedule

Obs	_TIME_	Engineer	Lengineer	Rengineer	Aengineer	Eengcost	Lengcost	Rengcost	Aengcost
1	01DEC03	2	2	2	1	0	0	0	40000
2	02DEC03	2	2	2	1	400	400	400	39600
3	03DEC03	2	2	2	1	800	800	800	39200
4	04DEC03	2	2	2	1	1200	1200	1200	38800
5	05DEC03	2	2	2	1	1600	1600	1600	38400
6	06DEC03	0	0	0	3	2000	2000	2000	38000
7	07DEC03	0	0	0	3	2000	2000	2000	38000
8	08DEC03	4	1	3	0	2000	2000	2000	38000
9	09DEC03	4	1	3	0	2800	2200	2600	37400
10	10DEC03	4	1	3	0	3600	2400	3200	36800
11	11DEC03	4	1	3	0	4400	2600	3800	36200
12	12DEC03	4	1	3	0	5200	2800	4400	35600
13	13DEC03	1	1	1	2	6000	3000	5000	35000
14	14DEC03	1	1	1	2	6000	3000	5000	35000
15	15DEC03	1	3	2	1	6000	3000	5000	35000
16	16DEC03	1	3	2	1	6200	3600	5400	34600
17	17DEC03	1	3	2	1	6400	4200	5800	34200
18	18DEC03	1	3	2	1	6600	4800	6200	33800
19	19DEC03	1	3	2	1	6800	5400	6600	33400
20	20DEC03	0	0	0	3	7000	6000	7000	33000
21	21DEC03	0	0	0	3	7000	6000	7000	33000
22	22DEC03	4	4	4	-1	7000	6000	7000	33000
23	23DEC03	4	4	4	-1	7800	6800	7800	32200
24	24DEC03	4	4	4	-1	8600	7600	8600	31400
25	25DEC03	4	4	4	-1	9400	8400	9400	30600
26	26DEC03	4	4	4	0	9400	8400	9400	30600
27	27DEC03	4	4	4	0	10200	9200	10200	29800
28	28DEC03	4	4	4	0	10200	9200	10200	29800
29	29DEC03	4	4	4	0	10200	9200	10200	29800
30	30DEC03	4	4	4	0	11000	10000	11000	29000
31	31DEC03	4	4	4	0	11800	10800	11800	28200
32	01JAN04	4	4	4	0	12600	11600	12600	27400
33	02JAN04	4	4	4	0	12600	11600	12600	27400
34	03JAN04	4	4	4	0	13400	12400	13400	26600
35	04JAN04	4	4	4	0	13400	12400	13400	26600
36	05JAN04	4	4	4	0	13400	12400	13400	26600
37	06JAN04	4	4	4	0	14200	13200	14200	25800
38	07JAN04	4	4	4	0	15000	14000	15000	25000
39	08JAN04	4	4	4	0	15800	14800	15800	24200
40	09JAN04	4	4	4	0	16600	15600	16600	23400

Output 4.16.2 *continued*

Using Supplementary Resources
Usage Profiles for Constrained Schedule

Obs	_TIME_	Engineer	Lengineer	Rengineer	Aengineer	Engcost	Lengcost	Rengcost	Aengcost
41	10JAN04	4	4	4	0	17400	16400	17400	22600
42	11JAN04	4	4	4	0	17400	16400	17400	22600
43	12JAN04	4	4	4	0	17400	16400	17400	22600
44	13JAN04	4	4	4	0	18200	17200	18200	21800
45	14JAN04	2	2	2	2	19000	18000	19000	21000
46	15JAN04	2	2	2	2	19400	18400	19400	20600
47	16JAN04	2	2	2	2	19800	18800	19800	20200
48	17JAN04	2	2	2	2	20200	19200	20200	19800
49	18JAN04	2	2	2	2	20200	19200	20200	19800
50	19JAN04	2	2	2	2	20200	19200	20200	19800
51	20JAN04	2	2	2	2	20600	19600	20600	19400
52	21JAN04	2	3	2	2	21000	20000	21000	19000
53	22JAN04	2	3	2	2	21400	20600	21400	18600
54	23JAN04	2	3	2	2	21800	21200	21800	18200
55	24JAN04	2	3	2	2	22200	21800	22200	17800
56	25JAN04	2	3	2	2	22200	21800	22200	17800
57	26JAN04	2	3	2	2	22200	21800	22200	17800
58	27JAN04	2	3	2	2	22600	22400	22600	17400
59	28JAN04	4	4	4	0	23000	23000	23000	17000
60	29JAN04	4	4	4	0	23800	23800	23800	16200
61	30JAN04	4	4	4	0	24600	24600	24600	15400
62	31JAN04	4	4	4	0	25400	25400	25400	14600
63	01FEB04	4	4	4	0	25400	25400	25400	14600
64	02FEB04	4	4	4	0	25400	25400	25400	14600
65	03FEB04	4	4	4	0	26200	26200	26200	13800
66	04FEB04	4	4	4	0	27000	27000	27000	13000
67	05FEB04	4	4	4	0	27800	27800	27800	12200
68	06FEB04	4	4	4	0	28600	28600	28600	11400
69	07FEB04	4	4	4	0	29400	29400	29400	10600
70	08FEB04	4	4	4	0	29400	29400	29400	10600
71	09FEB04	4	4	4	0	29400	29400	29400	10600
72	10FEB04	4	4	4	0	30200	30200	30200	9800
73	11FEB04	1	0	1	3	31000	31000	31000	9000
74	12FEB04	1	0	1	3	31200	31000	31200	8800
75	13FEB04	1	0	1	3	31400	31000	31400	8600
76	14FEB04	1	0	1	3	31600	31000	31600	8400
77	15FEB04	1	0	1	3	31600	31000	31600	8400
78	16FEB04	1	0	1	3	31600	31000	31600	8400
79	17FEB04	1	0	1	3	31800	31000	31800	8200
80	18FEB04	1	1	1	3	32000	31000	32000	8000

Output 4.16.2 *continued*

Using Supplementary Resources
Usage Profiles for Constrained Schedule

Obs	_TIME_	Eengineer	Lengineer	Rengineer	Aengineer	Eengcost	Lengcost	Rengcost	Aengcost
81	19FEB04	1	1	1	3	32200	31200	32200	7800
82	20FEB04	1	1	1	3	32400	31400	32400	7600
83	21FEB04	1	1	1	3	32600	31600	32600	7400
84	22FEB04	1	1	1	3	32600	31600	32600	7400
85	23FEB04	1	1	1	3	32600	31600	32600	7400
86	24FEB04	1	1	1	3	32800	31800	32800	7200
87	25FEB04	0	1	0	4	33000	32000	33000	7000
88	26FEB04	0	1	0	4	33000	32200	33000	7000
89	27FEB04	0	1	0	4	33000	32400	33000	7000
90	28FEB04	0	1	0	4	33000	32600	33000	7000
91	29FEB04	0	1	0	4	33000	32600	33000	7000
92	01MAR04	0	1	0	4	33000	32600	33000	7000
93	02MAR04	0	1	0	4	33000	32800	33000	7000
94	03MAR04	2	2	2	2	33000	33000	33000	7000
95	04MAR04	2	2	2	2	33400	33400	33400	6600
96	05MAR04	2	2	2	2	33800	33800	33800	6200
97	06MAR04	2	2	2	2	34200	34200	34200	5800
98	07MAR04	2	2	2	2	34200	34200	34200	5800
99	08MAR04	2	2	2	2	34200	34200	34200	5800
100	09MAR04	2	2	2	2	34600	34600	34600	5400
101	10MAR04	0	0	0	4	35000	35000	35000	5000

Example 4.17: INFEASDIAGNOSTIC Option and Aggregate Resource Type

The INFEASDIAGNOSTIC option instructs PROC CPM to continue scheduling even when resources are insufficient. When PROC CPM schedules subject to resource constraints, it stops the scheduling process when it cannot find sufficient resources (primary or supplementary) for an activity before the activity's latest possible start time (L_START + DELAY). In this case, you may want to determine which resources are needed to schedule all the activities and when the deficiencies occur. The INFEASDIAGNOSTIC option is equivalent to specifying infinite supplementary levels for all the resources under consideration; the DELAY= value is assumed to equal the default value of +INFINITY, unless it is specified otherwise.

The INFEASDIAGNOSTIC option is particularly useful when there are several resources involved and when project completion time is critical. You want things to be done on time, even if it means using supplementary resources or overtime resources; rather than trying to juggle activities around to try to fit available resource profiles, you want to determine the level of resources needed to accomplish tasks within a given time frame.

For the WIDGET manufacturing project, let us assume that there are four resources: a design engineer, a market analyst, a production engineer, and money. The resource requirements for the different activities are saved in a data set, WIDGR17, and displayed in Output 4.17.1. Of these resources, suppose that the design engineer is the resource that is most crucial in terms of his availability; perhaps he is an outside contractor and you do not have control over his availability. You need to determine the project schedule subject to the constraints on the resource deseng. Output 4.17.2 displays the RESOURCEIN= data set, RESIN17.

Output 4.17.1 Data Set WIDGR17

Use of the INFEASDIAGNOSTIC Option
Data Set WIDGR17

Obs	task	days	tail	head	deseng	mktan	prodeng	money
1	Approve Plan	5	1	2	1	1	1	200
2	Drawings	10	2	3	1	.	1	100
3	Study Market	5	2	4	.	1	1	100
4	Write Specs	5	2	3	1	.	1	150
5	Prototype	15	3	5	1	.	1	300
6	Mkt. Strat.	10	4	6	.	1	.	150
7	Materials	10	5	7	.	.	.	300
8	Facility	10	5	7	.	.	1	500
9	Init. Prod.	10	7	8	.	.	.	250
10	Evaluate	10	8	9	1	.	.	150
11	Test Market	15	6	9	.	1	.	200
12	Changes	5	9	10	1	.	1	200
13	Production	0	10	11	1	.	1	600
14	Marketing	0	6	12	.	1	.	.
15	Dummy	0	8	6	.	.	.	.

Output 4.17.2 Resourcein Data Set RESIN17

Use of the INFEASDIAGNOSTIC Option
Data Set RESIN17

Obs	per	otype	deseng	mktan	prodeng	money
1	.	restype	1	1	1	4
2	01DEC03	reslevel	1	.	1	.

In the first invocation of PROC CPM, the project is scheduled subject to resource constraints on the single resource variable deseng. Output 4.17.3 displays the resulting Schedule data set WIDGO17S, which shows that the project is delayed by five days because of this resource. The project finish time has been delayed only by five days, even though R_DELAY='10' for activity 'Write Specs'. This is due to the fact that there was a float of five days present in this activity.

```
proc cpm date='01dec03'd interval=weekday
         data=widgr17 holidata=holdata resin=resin17
         out=widgo17s;
   tailnode tail;
   duration days;
   headnode head;
```

```
holiday hol;
resource deseng / period=per obstype=otype
                  delayanalysis;
id task;
run;
```

Output 4.17.3 Resource-Constrained Schedule: Single Resource

Use of the INFEASDIAGNOSTIC Option
Resource Constrained Schedule: Single Resource

Obs	tail	head	days	task	deseng	S_START	S_FINISH	E_START	E_FINISH	L_START
1	1	2	5	Approve Plan	1	01DEC03	05DEC03	01DEC03	05DEC03	01DEC03
2	2	3	10	Drawings	1	08DEC03	19DEC03	08DEC03	19DEC03	08DEC03
3	2	4	5	Study Market	.	08DEC03	12DEC03	08DEC03	12DEC03	21JAN04
4	2	3	5	Write Specs	1	22DEC03	29DEC03	08DEC03	12DEC03	15DEC03
5	3	5	15	Prototype	1	30DEC03	20JAN04	22DEC03	13JAN04	22DEC03
6	4	6	10	Mkt. Strat.	.	15DEC03	29DEC03	15DEC03	29DEC03	28JAN04
7	5	7	10	Materials	.	21JAN04	03FEB04	14JAN04	27JAN04	14JAN04
8	5	7	10	Facility	.	21JAN04	03FEB04	14JAN04	27JAN04	14JAN04
9	7	8	10	Init. Prod.	.	04FEB04	17FEB04	28JAN04	10FEB04	28JAN04
10	8	9	10	Evaluate	1	18FEB04	02MAR04	11FEB04	24FEB04	18FEB04
11	6	9	15	Test Market	.	18FEB04	09MAR04	11FEB04	02MAR04	11FEB04
12	9	10	5	Changes	1	10MAR04	16MAR04	03MAR04	09MAR04	03MAR04
13	10	11	0	Production	1	17MAR04	17MAR04	10MAR04	10MAR04	10MAR04
14	6	12	0	Marketing	.	18FEB04	18FEB04	11FEB04	11FEB04	10MAR04
15	8	6	0	Dummy	.	18FEB04	18FEB04	11FEB04	11FEB04	11FEB04

Obs	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	05DEC03	0		
2	19DEC03	0		
3	27JAN04	0		
4	19DEC03	10	deseng	
5	13JAN04	0		
6	10FEB04	0		
7	27JAN04	0		
8	27JAN04	0		
9	10FEB04	0		
10	02MAR04	0		
11	02MAR04	0		
12	09MAR04	0		
13	10MAR04	0		
14	10MAR04	0		
15	11FEB04	0		

Now suppose that you have one production engineer available, but you could obtain more if needed. You do not want to delay the project more than five days (the delay caused by deseng). The second invocation of PROC CPM sets a maximum delay of five days on the activities and specifies all four resources along with the INFEASDIAGNOSTIC option. The resource availability data set (printed in Output 4.17.2) has missing values for the resources mktan and money. Further, the resource money is defined to be a consumable aggregate resource (its value is ‘4’ in the first observation). Thus, this resource is used by the CPM procedure only for aggregation purposes and is not considered as a constraining resource during the scheduling process. The INFEASDIAGNOSTIC option enables CPM to assume an infinite supplementary level for all the constraining resources, and the procedure draws upon this infinite reserve, if necessary, to schedule the project with only five days of delay. In other words, PROC CPM assumes that there is an infinite supply of supplementary levels for all the relevant resources. Thus, if at any point in the scheduling process it finds that an activity does not have enough resources and it cannot be postponed any further, it schedules the activity ignoring the insufficiency of the resources.

```
proc cpm date='01dec03'd interval=weekday
         data=widgr17 holidata=holdata resin=resin17
         out=widgo17m resout=widgro17;
   tailnode tail;
   duration days;
   headnode head;
   holiday hol;
   resource deseng prodeng mktan money / period=per obstype=otype
                                         delayanalysis
                                         delay=5
                                         infeasdiagnostic
                                         cumusage
                                         rcprofile avprofile;
   id task;
   run;
```

The Schedule data set WIDGO17M (for multiple resources) in Output 4.17.4 shows the new resource-constrained schedule. With a maximum delay of five days the procedure schedules the activity ‘Study Market’ on January 21, 2004, using an extra production engineer as indicated by the SUPPL_R variable. Note that the SUPPL_R variable indicates the first resource in the resource list that was used beyond its primary level. Note also that it is possible to schedule the activities with only one production engineer, but the project would be delayed by more than five days.

The Usage data set, displayed in Output 4.17.5, shows the amount of resources required on each day of the project. The data set contains usage and remaining resource information only for the resource-constrained schedule because PROC CPM was invoked with the RCPROFILE and AVPROFILE options in the RESOURCE statement. The availability profile contains only missing values for the resource money because it was used only for aggregation purposes. Further, since this resource is a consumable resource as per the RESOURCEIN= data set, and since the CUMUSAGE option is specified, the value for Rmoney in each observation indicates the cumulative amount of money that would be needed through the beginning of the date specified in that observation if the resource constrained schedule were followed.

For the other resources, the availability profile in the Usage data set contains negative values for all the resources that were insufficient on any given day. This feature is useful for diagnosing the level of insufficiency of any resource; you can determine the problem areas by examining the availability profile for the different resources. Thus, the negative values for the resource availability profile Aprodeng indicate that, in order for the project to be scheduled as desired, you need an extra production engineer between the 21st and 27th of January, 2004. The negative values for Amktan indicate the days when a market analyst is needed for the project.

Output 4.17.4 Resource-Constrained Schedule: Multiple Resources

Use of the INFEASDIAGNOSTIC Option
Resource Constrained Schedule: Multiple Resources

Obs	tail	head	days	task	deseng	prodeng	mktan	money	S_START	S_FINISH	E_START
1	1	2	5	Approve Plan	1	1	1	200	01DEC03	05DEC03	01DEC03
2	2	3	10	Drawings	1	1	.	100	08DEC03	19DEC03	08DEC03
3	2	4	5	Study Market	.	1	1	100	21JAN04	27JAN04	08DEC03
4	2	3	5	Write Specs	1	1	.	150	22DEC03	29DEC03	08DEC03
5	3	5	15	Prototype	1	1	.	300	30DEC03	20JAN04	22DEC03
6	4	6	10	Mkt. Strat.	.	.	1	150	28JAN04	10FEB04	15DEC03
7	5	7	10	Materials	.	.	.	300	21JAN04	03FEB04	14JAN04
8	5	7	10	Facility	.	1	.	500	21JAN04	03FEB04	14JAN04
9	7	8	10	Init. Prod.	.	.	.	250	04FEB04	17FEB04	28JAN04
10	8	9	10	Evaluate	1	.	.	150	18FEB04	02MAR04	11FEB04
11	6	9	15	Test Market	.	.	1	200	18FEB04	09MAR04	11FEB04
12	9	10	5	Changes	1	1	.	200	10MAR04	16MAR04	03MAR04
13	10	11	0	Production	1	1	.	600	17MAR04	17MAR04	10MAR04
14	6	12	0	Marketing	.	.	1	.	18FEB04	18FEB04	11FEB04
15	8	6	0	Dummy	.	.	.	.	18FEB04	18FEB04	11FEB04

Obs	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	05DEC03	01DEC03	05DEC03	0		mktan
2	19DEC03	08DEC03	19DEC03	0		
3	12DEC03	21JAN04	27JAN04	30	prodeng	prodeng
4	12DEC03	15DEC03	19DEC03	10	deseng	
5	13JAN04	22DEC03	13JAN04	0		
6	29DEC03	28JAN04	10FEB04	0		mktan
7	27JAN04	14JAN04	27JAN04	0		
8	27JAN04	14JAN04	27JAN04	0		
9	10FEB04	28JAN04	10FEB04	0		
10	24FEB04	18FEB04	02MAR04	0		
11	02MAR04	11FEB04	02MAR04	0		mktan
12	09MAR04	03MAR04	09MAR04	0		
13	10MAR04	10MAR04	10MAR04	0		
14	11FEB04	10MAR04	10MAR04	0		
15	11FEB04	11FEB04	11FEB04	0		

Output 4.17.5 Resource Usage: Multiple Resources

Use of the INFEASDIAGNOSTIC Option
Usage Profile: Multiple Resources

Obs	_TIME_	Rdeseng	Adeseng	Rprodeng	Aprodeng	Rmktan	Amktan	Rmoney	Amoney
1	01DEC03	1	0	1	0	1	-1	0	.
2	02DEC03	1	0	1	0	1	-1	200	.
3	03DEC03	1	0	1	0	1	-1	400	.
4	04DEC03	1	0	1	0	1	-1	600	.
5	05DEC03	1	0	1	0	1	-1	800	.
6	08DEC03	1	0	1	0	0	0	1000	.
7	09DEC03	1	0	1	0	0	0	1100	.
8	10DEC03	1	0	1	0	0	0	1200	.
9	11DEC03	1	0	1	0	0	0	1300	.
10	12DEC03	1	0	1	0	0	0	1400	.
11	15DEC03	1	0	1	0	0	0	1500	.
12	16DEC03	1	0	1	0	0	0	1600	.
13	17DEC03	1	0	1	0	0	0	1700	.
14	18DEC03	1	0	1	0	0	0	1800	.
15	19DEC03	1	0	1	0	0	0	1900	.
16	22DEC03	1	0	1	0	0	0	2000	.
17	23DEC03	1	0	1	0	0	0	2150	.
18	24DEC03	1	0	1	0	0	0	2300	.
19	26DEC03	1	0	1	0	0	0	2450	.
20	29DEC03	1	0	1	0	0	0	2600	.
21	30DEC03	1	0	1	0	0	0	2750	.
22	31DEC03	1	0	1	0	0	0	3050	.
23	02JAN04	1	0	1	0	0	0	3350	.
24	05JAN04	1	0	1	0	0	0	3650	.
25	06JAN04	1	0	1	0	0	0	3950	.
26	07JAN04	1	0	1	0	0	0	4250	.
27	08JAN04	1	0	1	0	0	0	4550	.
28	09JAN04	1	0	1	0	0	0	4850	.
29	12JAN04	1	0	1	0	0	0	5150	.
30	13JAN04	1	0	1	0	0	0	5450	.
31	14JAN04	1	0	1	0	0	0	5750	.
32	15JAN04	1	0	1	0	0	0	6050	.
33	16JAN04	1	0	1	0	0	0	6350	.
34	19JAN04	1	0	1	0	0	0	6650	.
35	20JAN04	1	0	1	0	0	0	6950	.
36	21JAN04	0	1	2	-1	1	-1	7250	.
37	22JAN04	0	1	2	-1	1	-1	8150	.
38	23JAN04	0	1	2	-1	1	-1	9050	.
39	26JAN04	0	1	2	-1	1	-1	9950	.
40	27JAN04	0	1	2	-1	1	-1	10850	.
41	28JAN04	0	1	1	0	1	-1	11750	.
42	29JAN04	0	1	1	0	1	-1	12700	.
43	30JAN04	0	1	1	0	1	-1	13650	.
44	02FEB04	0	1	1	0	1	-1	14600	.
45	03FEB04	0	1	1	0	1	-1	15550	.
46	04FEB04	0	1	0	1	1	-1	16500	.

Output 4.17.5 *continued*

Use of the INFEASDIAGNOSTIC Option
Usage Profile: Multiple Resources

Obs	_TIME_	Rdeseng	Adeseng	Rprodeng	Aprodeng	Rmktan	Amktan	Rmoney	Amoney
47	05FEB04	0	1	0	1	1	-1	16900	.
48	06FEB04	0	1	0	1	1	-1	17300	.
49	09FEB04	0	1	0	1	1	-1	17700	.
50	10FEB04	0	1	0	1	1	-1	18100	.
51	11FEB04	0	1	0	1	0	0	18500	.
52	12FEB04	0	1	0	1	0	0	18750	.
53	13FEB04	0	1	0	1	0	0	19000	.
54	16FEB04	0	1	0	1	0	0	19250	.
55	17FEB04	0	1	0	1	0	0	19500	.
56	18FEB04	1	0	0	1	1	-1	19750	.
57	19FEB04	1	0	0	1	1	-1	20100	.
58	20FEB04	1	0	0	1	1	-1	20450	.
59	23FEB04	1	0	0	1	1	-1	20800	.
60	24FEB04	1	0	0	1	1	-1	21150	.
61	25FEB04	1	0	0	1	1	-1	21500	.
62	26FEB04	1	0	0	1	1	-1	21850	.
63	27FEB04	1	0	0	1	1	-1	22200	.
64	01MAR04	1	0	0	1	1	-1	22550	.
65	02MAR04	1	0	0	1	1	-1	22900	.
66	03MAR04	0	1	0	1	1	-1	23250	.
67	04MAR04	0	1	0	1	1	-1	23450	.
68	05MAR04	0	1	0	1	1	-1	23650	.
69	08MAR04	0	1	0	1	1	-1	23850	.
70	09MAR04	0	1	0	1	1	-1	24050	.
71	10MAR04	1	0	1	0	0	0	24250	.
72	11MAR04	1	0	1	0	0	0	24450	.
73	12MAR04	1	0	1	0	0	0	24650	.
74	15MAR04	1	0	1	0	0	0	24850	.
75	16MAR04	1	0	1	0	0	0	25050	.
76	17MAR04	0	1	0	1	0	0	25250	.

Example 4.18: Variable Activity Delay

In Example 4.17, the DELAY= option is used to specify a maximum amount of delay that is allowed for all activities in the project. In some situations it may be reasonable to set the delay for each activity based on some characteristic pertaining to the activity. For example, consider the data in Example 4.17 with a slightly different scenario. Suppose that no delay is allowed in activities that require a production engineer. Data set WIDGR18, displayed in Output 4.18.1, is obtained from WIDGR17 using the following simple DATA step.

```
data widgr18;
   set widgr17;
   if prodeng ^= . then adelay = 0;
   else                 adelay = 5;
   run;

title 'Variable Activity Delay';
title2 'Data Set WIDGR18';
proc print;
   run;
```

Output 4.18.1 Activity Data Set WIDGR18

Variable Activity Delay
Data Set WIDGR18

Obs	task	days	tail	head	deseng	mktan	prodeng	money	adelay
1	Approve Plan	5	1	2	1	1	1	200	0
2	Drawings	10	2	3	1	.	1	100	0
3	Study Market	5	2	4	.	1	1	100	0
4	Write Specs	5	2	3	1	.	1	150	0
5	Prototype	15	3	5	1	.	1	300	0
6	Mkt. Strat.	10	4	6	.	1	.	150	5
7	Materials	10	5	7	.	.	.	300	5
8	Facility	10	5	7	.	.	1	500	0
9	Init. Prod.	10	7	8	.	.	.	250	5
10	Evaluate	10	8	9	1	.	.	150	5
11	Test Market	15	6	9	.	1	.	200	5
12	Changes	5	9	10	1	.	1	200	0
13	Production	0	10	11	1	.	1	600	0
14	Marketing	0	6	12	.	1	.	.	5
15	Dummy	0	8	6	.	.	.	.	5

PROC CPM is invoked with the ACTDELAY=ADELAY option in the RESOURCE statement. The INFEASDIAGNOSTIC option is also used to enable the procedure to schedule activities even if resources are insufficient. The output data sets are displayed in Output 4.18.2 and Output 4.18.3.

```
data resin17;
   input per & date7. otype $
         deseng mktan prodeng money;
   format per date7.;
   datalines;
.        restype  1 1 1 4
01dec03  reslevel 1 . 1 .
;

data holdata;
   format hol date7. name $9. ;
   input hol & date7. name & ;
   datalines;
25dec03  Christmas
01jan04  New Year
;

proc cpm date='01dec03'd
         interval=weekday
         data=widgr18
         holidata=holdata
         resin=resin17
         out=widgo18
         resout=widgro18;
   tailnode tail;
   duration days;
   headnode head;
   holiday hol;
   resource deseng prodeng mktan money / period=per
                                         obstype=otype
                                         delayanalysis
                                         actdelay=adelay
                                         infeasdiagnostic
                                         rcs avl t_float
                                         cumusage;
   id task;
   run;
```

Output 4.18.2 Resource-Constrained Schedule: Variable Activity Delay

Variable Activity Delay
Resource Constrained Schedule

Obs	tail	head	days	task	adelay	deseng	prodeng	mktan	money	S_START	S_FINISH	E_START
1	1	2	5	Approve Plan	0	1	1	1	200	01DEC03	05DEC03	01DEC03
2	2	3	10	Drawings	0	1	1	.	100	08DEC03	19DEC03	08DEC03
3	2	4	5	Study Market	0	.	1	1	100	14JAN04	20JAN04	08DEC03
4	2	3	5	Write Specs	0	1	1	.	150	08DEC03	12DEC03	08DEC03
5	3	5	15	Prototype	0	1	1	.	300	22DEC03	13JAN04	22DEC03
6	4	6	10	Mkt. Strat.	5	.	.	1	150	21JAN04	03FEB04	15DEC03
7	5	7	10	Materials	5	.	.	.	300	14JAN04	27JAN04	14JAN04
8	5	7	10	Facility	0	.	1	.	500	14JAN04	27JAN04	14JAN04
9	7	8	10	Init. Prod.	5	.	.	.	250	28JAN04	10FEB04	28JAN04
10	8	9	10	Evaluate	5	1	.	.	150	11FEB04	24FEB04	11FEB04
11	6	9	15	Test Market	5	.	.	1	200	11FEB04	02MAR04	11FEB04
12	9	10	5	Changes	0	1	1	.	200	03MAR04	09MAR04	03MAR04
13	10	11	0	Production	0	1	1	.	600	10MAR04	10MAR04	10MAR04
14	6	12	0	Marketing	5	.	.	1	.	11FEB04	11FEB04	11FEB04
15	8	6	0	Dummy	5	.	.	.	.	11FEB04	11FEB04	11FEB04

Obs	E_FINISH	L_START	L_FINISH	T_FLOAT	R_DELAY	DELAY_R	SUPPL_R
1	05DEC03	01DEC03	05DEC03	0	0		mktan
2	19DEC03	08DEC03	19DEC03	0	0		
3	12DEC03	21JAN04	27JAN04	30	25	prodeng	prodeng
4	12DEC03	15DEC03	19DEC03	5	0		deseng
5	13JAN04	22DEC03	13JAN04	0	0		
6	29DEC03	28JAN04	10FEB04	30	0		mktan
7	27JAN04	14JAN04	27JAN04	0	0		
8	27JAN04	14JAN04	27JAN04	0	0		
9	10FEB04	28JAN04	10FEB04	0	0		
10	24FEB04	18FEB04	02MAR04	5	0		
11	02MAR04	11FEB04	02MAR04	0	0		mktan
12	09MAR04	03MAR04	09MAR04	0	0		
13	10MAR04	10MAR04	10MAR04	0	0		
14	11FEB04	10MAR04	10MAR04	20	0		
15	11FEB04	11FEB04	11FEB04	0	0		

Output 4.18.3 Resource Usage

Variable Activity Delay
Usage Profile

Obs	_TIME_	Rdeseng	Adeseng	Rprodeng	Aprodeng	Rmktan	Amktan	Rmoney	Amoney
1	01DEC03	1	0	1	0	1	-1	0	.
2	02DEC03	1	0	1	0	1	-1	200	.
3	03DEC03	1	0	1	0	1	-1	400	.
4	04DEC03	1	0	1	0	1	-1	600	.
5	05DEC03	1	0	1	0	1	-1	800	.
6	08DEC03	2	-1	2	-1	0	0	1000	.
7	09DEC03	2	-1	2	-1	0	0	1250	.
8	10DEC03	2	-1	2	-1	0	0	1500	.
9	11DEC03	2	-1	2	-1	0	0	1750	.
10	12DEC03	2	-1	2	-1	0	0	2000	.
11	15DEC03	1	0	1	0	0	0	2250	.
12	16DEC03	1	0	1	0	0	0	2350	.
13	17DEC03	1	0	1	0	0	0	2450	.
14	18DEC03	1	0	1	0	0	0	2550	.
15	19DEC03	1	0	1	0	0	0	2650	.
16	22DEC03	1	0	1	0	0	0	2750	.
17	23DEC03	1	0	1	0	0	0	3050	.
18	24DEC03	1	0	1	0	0	0	3350	.
19	26DEC03	1	0	1	0	0	0	3650	.
20	29DEC03	1	0	1	0	0	0	3950	.
21	30DEC03	1	0	1	0	0	0	4250	.
22	31DEC03	1	0	1	0	0	0	4550	.
23	02JAN04	1	0	1	0	0	0	4850	.
24	05JAN04	1	0	1	0	0	0	5150	.
25	06JAN04	1	0	1	0	0	0	5450	.
26	07JAN04	1	0	1	0	0	0	5750	.
27	08JAN04	1	0	1	0	0	0	6050	.
28	09JAN04	1	0	1	0	0	0	6350	.
29	12JAN04	1	0	1	0	0	0	6650	.
30	13JAN04	1	0	1	0	0	0	6950	.
31	14JAN04	0	1	2	-1	1	-1	7250	.
32	15JAN04	0	1	2	-1	1	-1	8150	.
33	16JAN04	0	1	2	-1	1	-1	9050	.
34	19JAN04	0	1	2	-1	1	-1	9950	.
35	20JAN04	0	1	2	-1	1	-1	10850	.
36	21JAN04	0	1	1	0	1	-1	11750	.
37	22JAN04	0	1	1	0	1	-1	12700	.
38	23JAN04	0	1	1	0	1	-1	13650	.
39	26JAN04	0	1	1	0	1	-1	14600	.
40	27JAN04	0	1	1	0	1	-1	15550	.
41	28JAN04	0	1	0	1	1	-1	16500	.
42	29JAN04	0	1	0	1	1	-1	16900	.
43	30JAN04	0	1	0	1	1	-1	17300	.
44	02FEB04	0	1	0	1	1	-1	17700	.
45	03FEB04	0	1	0	1	1	-1	18100	.
46	04FEB04	0	1	0	1	0	0	18500	.

Output 4.18.3 *continued*

Variable Activity Delay
Usage Profile

Obs	_TIME_	Rdeseng	Adeseng	Rprodeng	Aprodeng	Rmktan	Amktan	Rmoney	Amoney
47	05FEB04	0	1	0	1	0	0	18750	.
48	06FEB04	0	1	0	1	0	0	19000	.
49	09FEB04	0	1	0	1	0	0	19250	.
50	10FEB04	0	1	0	1	0	0	19500	.
51	11FEB04	1	0	0	1	1	-1	19750	.
52	12FEB04	1	0	0	1	1	-1	20100	.
53	13FEB04	1	0	0	1	1	-1	20450	.
54	16FEB04	1	0	0	1	1	-1	20800	.
55	17FEB04	1	0	0	1	1	-1	21150	.
56	18FEB04	1	0	0	1	1	-1	21500	.
57	19FEB04	1	0	0	1	1	-1	21850	.
58	20FEB04	1	0	0	1	1	-1	22200	.
59	23FEB04	1	0	0	1	1	-1	22550	.
60	24FEB04	1	0	0	1	1	-1	22900	.
61	25FEB04	0	1	0	1	1	-1	23250	.
62	26FEB04	0	1	0	1	1	-1	23450	.
63	27FEB04	0	1	0	1	1	-1	23650	.
64	01MAR04	0	1	0	1	1	-1	23850	.
65	02MAR04	0	1	0	1	1	-1	24050	.
66	03MAR04	1	0	1	0	0	0	24250	.
67	04MAR04	1	0	1	0	0	0	24450	.
68	05MAR04	1	0	1	0	0	0	24650	.
69	08MAR04	1	0	1	0	0	0	24850	.
70	09MAR04	1	0	1	0	0	0	25050	.
71	10MAR04	0	1	0	1	0	0	25250	.

Note from the Schedule data set that the activity 'Study Market' is scheduled to start on January 14, 2004, even though (L_START + adelay)=21JAN04. This is due to the fact that at every time interval, the scheduling algorithm looks ahead in time to detect any increase in the primary level of the resource; if the future resource profile indicates that the procedure will need to use supplementary levels anyway, the activity will not be forced to wait until (L_START + DELAY). (To force the activity to wait until its latest allowed start time, use the AWAITDELAY option). The DELAYANALYSIS variables indicate that a supplementary level of the resource prodeng is needed to schedule the activity on 14JAN03. The variable SUPPL_R identifies only one supplementary resource that is needed for the activity. In fact, examination of the resource requirements for the activity and the RESOURCEOUT data set shows that an extra market analyst is also needed between the 14th and 20th of January to schedule this activity. Likewise, the activities 'Write Specs' and 'Drawings' require a design engineer and a production engineer; both these activities start on the 8th of December. The RESOURCEOUT data set indicates that an extra design engineer and an extra production engineer are needed from the 8th to the 12th of December.

The next invocation of PROC CPM illustrates the use of the ACTDELAY variable to force the resource-constrained schedule to coincide with the early start schedule. The following DATA step uses the Schedule data set WIDGO18 to set an activity delay variable (actdel) to be equal to −T_FLOAT. PROC CPM is then invoked with the ACTDELAY variable equal to actdel and the INFEASDIAGNOSTIC option. This forces all activities to be scheduled on or before (L_START + actdel), which happens to be equal to E_START; thus all activities are scheduled to start at their early start time. The resulting Schedule data set is displayed in Output 4.18.4. Though this is an extreme case, a similar technique could be used selectively to set the delay value for each activity (or some of the activities) to depend on the unconstrained schedule or the T_FLOAT value. If both the DELAY= and ACTDELAY= options are specified, the DELAY= value is used to set the activity delay values for activities that have missing values for the ACTDELAY variable.

Note also that in this invocation of PROC CPM, the BASELINE statement is used to compare the early start schedule and the resource constrained schedule. The S_VAR and F_VAR variables are 0 for all the activities, as is to be expected (since all activities are forced to start as per the early start schedule).

```
data negdelay;
   set widgo18;
   actdel=-t_float;
   run;

proc cpm date='01dec03'd
         interval=weekday
         data=negdelay
         holidata=holdata
         resin=resin17
         out=widgo18n;
   tailnode tail;
   duration days;
   headnode head;
   holiday hol;
   resource deseng prodeng mktan money / period=per
                                         obstype=otype
                                         delayanalysis
                                         actdelay=actdel
                                         infeasdiagnostic;
   baseline / set=early compare=resource;
   id task;
   run;
```

Output 4.18.4 Resource-Constrained Schedule: Activity Delay = - (T_FLOAT)

Variable Activity Delay
Resource Constrained Schedule
Activity Delay = - (T_FLOAT)

Obs	tail	head	days	task	actdel	deseng	prodeng	mktan	money	S_START	S_FINISH	E_START
1	1	2	5	Approve Plan	0	1	1	1	200	01DEC03	05DEC03	01DEC03
2	2	3	10	Drawings	0	1	1	.	100	08DEC03	19DEC03	08DEC03
3	2	4	5	Study Market	-30	.	1	1	100	08DEC03	12DEC03	08DEC03
4	2	3	5	Write Specs	-5	1	1	.	150	08DEC03	12DEC03	08DEC03
5	3	5	15	Prototype	0	1	1	.	300	22DEC03	13JAN04	22DEC03
6	4	6	10	Mkt. Strat.	-30	.	.	1	150	15DEC03	29DEC03	15DEC03
7	5	7	10	Materials	0	.	.	.	300	14JAN04	27JAN04	14JAN04
8	5	7	10	Facility	0	.	1	.	500	14JAN04	27JAN04	14JAN04
9	7	8	10	Init. Prod.	0	.	.	.	250	28JAN04	10FEB04	28JAN04
10	8	9	10	Evaluate	-5	1	.	.	150	11FEB04	24FEB04	11FEB04
11	6	9	15	Test Market	0	.	.	1	200	11FEB04	02MAR04	11FEB04
12	9	10	5	Changes	0	1	1	.	200	03MAR04	09MAR04	03MAR04
13	10	11	0	Production	0	1	1	.	600	10MAR04	10MAR04	10MAR04
14	6	12	0	Marketing	-20	.	.	1	.	11FEB04	11FEB04	11FEB04
15	8	6	0	Dummy	0	.	.	.	.	11FEB04	11FEB04	11FEB04

Obs	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R	B_START	B_FINISH	S_VAR	F_VAR
1	05DEC03	01DEC03	05DEC03	0		mktan	01DEC03	05DEC03	0	0
2	19DEC03	08DEC03	19DEC03	0			08DEC03	19DEC03	0	0
3	12DEC03	21JAN04	27JAN04	0		prodeng	08DEC03	12DEC03	0	0
4	12DEC03	15DEC03	19DEC03	0		deseng	08DEC03	12DEC03	0	0
5	13JAN04	22DEC03	13JAN04	0			22DEC03	13JAN04	0	0
6	29DEC03	28JAN04	10FEB04	0		mktan	15DEC03	29DEC03	0	0
7	27JAN04	14JAN04	27JAN04	0			14JAN04	27JAN04	0	0
8	27JAN04	14JAN04	27JAN04	0			14JAN04	27JAN04	0	0
9	10FEB04	28JAN04	10FEB04	0			28JAN04	10FEB04	0	0
10	24FEB04	18FEB04	02MAR04	0			11FEB04	24FEB04	0	0
11	02MAR04	11FEB04	02MAR04	0		mktan	11FEB04	02MAR04	0	0
12	09MAR04	03MAR04	09MAR04	0			03MAR04	09MAR04	0	0
13	10MAR04	10MAR04	10MAR04	0			10MAR04	10MAR04	0	0
14	11FEB04	10MAR04	10MAR04	0			11FEB04	11FEB04	0	0
15	11FEB04	11FEB04	11FEB04	0			11FEB04	11FEB04	0	0

Example 4.19: Activity Splitting

This example illustrates the use of activity splitting to help reduce project duration. By default, PROC CPM assumes that an activity cannot be interrupted once it is started (except for holidays and weekends). During resource-constrained scheduling, it is possible for a noncritical activity to be scheduled first, and at a later time a critical activity may be held waiting for a resource to be freed by this less critical activity. In such situations, you way want to allow noncritical activities to be preempted by critical ones. PROC CPM enables you to specify, selectively, the activities that can be split into segments, the minimum length of each segment, and the maximum number of segments per activity.

The data set WIDGR19, displayed in Output 4.19.1, contains the widget network in AON format with two resources: prodman and hrdware. Suppose the production manager is required to oversee certain activities, as indicated by a '1' in the prodman column. hrdware denotes some piece of equipment that is required by the activity 'Drawings' (perhaps a plotter to produce the engineering drawings). The variable minseg denotes the minimum length of the split segments for each activity. Missing values for this variable are set to default values (one-fifth of the activity's duration). The Resource data set WIDGRIN, displayed in Output 4.19.2, indicates that both resources are replenishable, there is one production manager available from December 1, and the hardware is unavailable on the 10th and 11th of December (perhaps it is scheduled for maintenance or has been reserved for some other project).

Output 4.19.1 Activity Splitting: Activity Data Set

Activity Splitting
Project Data

Obs	task	days	succ	prodman	hrdware	minseg
1	Approve Plan	5	Drawings	1	.	.
2	Approve Plan	5	Study Market	1	.	.
3	Approve Plan	5	Write Specs	1	.	.
4	Drawings	10	Prototype	.	1	1
5	Study Market	5	Mkt. Strat.	.	.	.
6	Write Specs	5	Prototype	.	.	.
7	Prototype	15	Materials	1	.	.
8	Prototype	15	Facility	1	.	.
9	Mkt. Strat.	10	Test Market	1	.	1
10	Mkt. Strat.	10	Marketing	1	.	1
11	Materials	10	Init. Prod.	.	.	.
12	Facility	10	Init. Prod.	.	.	.
13	Init. Prod.	10	Test Market	1	.	.
14	Init. Prod.	10	Marketing	1	.	.
15	Init. Prod.	10	Evaluate	1	.	.
16	Evaluate	10	Changes	1	.	.
17	Test Market	15	Changes	.	.	.
18	Changes	5	Production	.	.	.
19	Production	0		1	.	.
20	Marketing	0		.	.	.

Output 4.19.2 Activity Splitting: Resource Availability Data Set

Activity Splitting
Resource Availability Data Set

Obs	per	otype	prodman	hrdware
1	.	restype	1	1
2	01DEC03	reslevel	1	1
3	10DEC03	reslevel	.	0
4	12DEC03	reslevel	.	1

The project is first scheduled without allowing any of the activities to be split. The Schedule data set SCHED, displayed in Output 4.19.3, indicates that the project has been delayed by one week (five working days, since maximum S_FINISH = '17MAR04' while maximum E_FINISH = '10MAR04'). The activity 'Drawings' has been postponed to start after the equipment has been serviced (or used by the other project), and the activity 'Prototype' (which is actually a critical activity) cannot start on schedule because the production manager is tied up with the noncritical activity 'Mkt. Strat.'.

```
proc cpm date='01dec03'd
         data=widgr19 resin=widgrin
         holidata=holdata
         out=sched resout=rout
         interval=weekday collapse;
   activity task;
   duration days;
   successor succ;
   holiday hol;
   resource prodman hrdware / period=per obstype=otype
                              t_float f_float rcs avl;
   run;
```

Output 4.19.3 Project Schedule: Splitting Not Allowed

Activity Splitting
Project Schedule: Splitting not Allowed

Obs	task	succ	days	prodman	hrdware	S_START	S_FINISH	E_START	E_FINISH
1	Approve Plan	Drawings	5	1	.	01DEC03	05DEC03	01DEC03	05DEC03
2	Drawings	Prototype	10	.	1	12DEC03	26DEC03	08DEC03	19DEC03
3	Study Market	Mkt. Strat.	5	.	.	08DEC03	12DEC03	08DEC03	12DEC03
4	Write Specs	Prototype	5	.	.	08DEC03	12DEC03	08DEC03	12DEC03
5	Prototype	Materials	15	1	.	30DEC03	20JAN04	22DEC03	13JAN04
6	Mkt. Strat.	Test Market	10	1	.	15DEC03	29DEC03	15DEC03	29DEC03
7	Materials	Init. Prod.	10	.	.	21JAN04	03FEB04	14JAN04	27JAN04
8	Facility	Init. Prod.	10	.	.	21JAN04	03FEB04	14JAN04	27JAN04
9	Init. Prod.	Test Market	10	1	.	04FEB04	17FEB04	28JAN04	10FEB04
10	Evaluate	Changes	10	1	.	18FEB04	02MAR04	11FEB04	24FEB04
11	Test Market	Changes	15	.	.	18FEB04	09MAR04	11FEB04	02MAR04
12	Changes	Production	5	.	.	10MAR04	16MAR04	03MAR04	09MAR04
13	Production		0	1	.	17MAR04	17MAR04	10MAR04	10MAR04
14	Marketing		0	.	.	18FEB04	18FEB04	11FEB04	11FEB04

Obs	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	01DEC03	05DEC03	0	0
2	08DEC03	19DEC03	0	0
3	21JAN04	27JAN04	30	0
4	15DEC03	19DEC03	5	5
5	22DEC03	13JAN04	0	0
6	28JAN04	10FEB04	30	30
7	14JAN04	27JAN04	0	0
8	14JAN04	27JAN04	0	0
9	28JAN04	10FEB04	0	0
10	18FEB04	02MAR04	5	5
11	11FEB04	02MAR04	0	0
12	03MAR04	09MAR04	0	0
13	10MAR04	10MAR04	0	0
14	10MAR04	10MAR04	20	20

In the second invocation of PROC CPM, the MINSEGMTDUR= option is used in the RESOURCE statement to identify the variable minseg to the procedure. This enables the algorithm to split the ‘Drawings’ activity so that some of it is done before December 10, 2003, and the rest is scheduled to start on December 12, 2003. Likewise, the production manager is allocated to the activity ‘Mkt. Strat.’ on December 15, 2003. On the 24th of December the activity ‘Prototype’ demands the production manager, and since preemption is allowed, the earlier activity ‘Mkt. Strat.’, which is less critical than ‘Prototype’, is temporarily halted and is resumed on the 16th of January after the completion of ‘Prototype’ on the 15th of January. The Schedule data set, displayed in Output 4.19.4, contains separate observations for each segment of the split activities as indicated by the variable SEGMT_NO. The project duration has been reduced by three working days, by allowing appropriate activities to be split.

```
proc cpm date='01dec03'd
         data=widgr19
         holidata=holdata resin=widgrin
         out=spltschd resout=spltrout
         interval=weekday collapse;
   activity task;
   duration days;
   successor succ;
   holiday hol;
   resource prodman hrdware / period=per obstype=otype
                              minsegmtdur=minseg
                              rcs avl;
   id task;
   run;
```

Output 4.19.4 Project Schedule: Splitting Allowed

Activity Splitting
Project Schedule: Splitting Allowed

Obs	task	succ	SEGMT_NO	days	prodman	hrdware
1	Approve Plan	Drawings	.	5	1	.
2	Drawings	Prototype	.	10	.	1
3	Drawings	Prototype	1	2	.	1
4	Drawings	Prototype	2	8	.	1
5	Study Market	Mkt. Strat.	.	5	.	.
6	Write Specs	Prototype	.	5	.	.
7	Prototype	Materials	.	15	1	.
8	Mkt. Strat.	Test Market	.	10	1	.
9	Mkt. Strat.	Test Market	1	7	1	.
10	Mkt. Strat.	Test Market	2	3	1	.
11	Materials	Init. Prod.	.	10	.	.
12	Facility	Init. Prod.	.	10	.	.
13	Init. Prod.	Test Market	.	10	1	.
14	Evaluate	Changes	.	10	1	.
15	Test Market	Changes	.	15	.	.
16	Changes	Production	.	5	.	.
17	Production		.	0	1	.
18	Marketing		.	0	.	.

Obs	S_START	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
1	01DEC03	05DEC03	01DEC03	05DEC03	01DEC03	05DEC03
2	08DEC03	23DEC03	08DEC03	19DEC03	08DEC03	19DEC03
3	08DEC03	09DEC03	08DEC03	19DEC03	08DEC03	19DEC03
4	12DEC03	23DEC03	08DEC03	19DEC03	08DEC03	19DEC03
5	08DEC03	12DEC03	08DEC03	12DEC03	21JAN04	27JAN04
6	08DEC03	12DEC03	08DEC03	12DEC03	15DEC03	19DEC03
7	24DEC03	15JAN04	22DEC03	13JAN04	22DEC03	13JAN04
8	15DEC03	20JAN04	15DEC03	29DEC03	28JAN04	10FEB04
9	15DEC03	23DEC03	15DEC03	29DEC03	28JAN04	10FEB04
10	16JAN04	20JAN04	15DEC03	29DEC03	28JAN04	10FEB04
11	16JAN04	29JAN04	14JAN04	27JAN04	14JAN04	27JAN04
12	16JAN04	29JAN04	14JAN04	27JAN04	14JAN04	27JAN04
13	30JAN04	12FEB04	28JAN04	10FEB04	28JAN04	10FEB04
14	13FEB04	26FEB04	11FEB04	24FEB04	18FEB04	02MAR04
15	13FEB04	04MAR04	11FEB04	02MAR04	11FEB04	02MAR04
16	05MAR04	11MAR04	03MAR04	09MAR04	03MAR04	09MAR04
17	12MAR04	12MAR04	10MAR04	10MAR04	10MAR04	10MAR04
18	13FEB04	13FEB04	11FEB04	11FEB04	10MAR04	10MAR04

Example 4.20: Alternate Resources

Some projects may have two or more resource types that are interchangeable; if one resource is insufficient, the other one can be used in its place. To illustrate the use of alternate resources, consider the widget manufacturing example with the data in AON format as shown in Output 4.20.1. As in Example 4.17, suppose there are two types of engineers, a design engineer and a production engineer. In addition, there is a generic pool of engineers, denoted by the variable engpool. The resource requirements for each category are specified in the data set WIDGR20.

Output 4.20.1 Alternate Resources: Activity Data Set

Scheduling with Alternate Resources
Data Set WIDGR20

Obs	task	days	succ	deseng	prodeng	engpool
1	Approve Plan	5	Drawings	1	1	.
2	Approve Plan	5	Study Market	1	1	.
3	Approve Plan	5	Write Specs	1	1	.
4	Drawings	10	Prototype	1	1	.
5	Study Market	5	Mkt. Strat.	.	1	.
6	Write Specs	5	Prototype	1	1	.
7	Prototype	15	Materials	1	1	1
8	Prototype	15	Facility	1	1	1
9	Mkt. Strat.	10	Test Market	.	.	.
10	Mkt. Strat.	10	Marketing	.	.	.
11	Materials	10	Init. Prod.	.	.	.
12	Facility	10	Init. Prod.	.	1	2
13	Init. Prod.	10	Test Market	.	.	2
14	Init. Prod.	10	Marketing	.	.	2
15	Init. Prod.	10	Evaluate	.	.	2
16	Evaluate	10	Changes	1	.	.
17	Test Market	15	Changes	.	.	.
18	Changes	5	Production	1	1	.
19	Production	0		.	.	.
20	Marketing	0		.	.	.

Output 4.20.2 Alternate Resources: RESOURCEIN Data Set

Scheduling with Alternate Resources
Data Set RESIN20

Obs	per	otype	resid	deseng	prodeng	engpool
1	.	restype		1	1	1
2	.	altprty	deseng	.	1	2
3	.	altprty	prodeng	.	.	1
4	.	suplevel		1	1	.
5	01DEC03	reslevel		1	1	4

The resource availability data set RESIN20, displayed in Output 4.20.2, identifies all three resources as replenishable resources and indicates a primary as well as a supplementary level of availability. A new variable resid in the data set is used to identify resources in observations 2 and 3 that can be substituted for deseng and prodeng, respectively. These observations have the value 'altprty' for the OBSTYPE variable and indicate a priority for the substitution. For example, observation number 2 indicates that if deseng is unavailable, the procedure can use prodeng, and if even that is insufficient, it can draw from the engineering resource pool engpool. To trigger the substitution of resources, use the RESID= option in the RESOURCE statement.

Output 4.20.3 Alternate Resources Not Used

Scheduling with Alternate Resources
Alternate Resources not used

Obs	task	succ	days	deseng	prodeng	engpool	S_START	S_FINISH
1	Approve Plan	Drawings	5	1	1	.	01DEC03	05DEC03
2	Drawings	Prototype	10	1	1	.	08DEC03	19DEC03
3	Study Market	Mkt. Strat.	5	.	1	.	04FEB04	10FEB04
4	Write Specs	Prototype	5	1	1	.	22DEC03	29DEC03
5	Prototype	Materials	15	1	1	1	30DEC03	20JAN04
6	Mkt. Strat.	Test Market	10	.	.	.	11FEB04	24FEB04
7	Materials	Init. Prod.	10	.	.	.	21JAN04	03FEB04
8	Facility	Init. Prod.	10	.	1	2	21JAN04	03FEB04
9	Init. Prod.	Test Market	10	.	.	2	04FEB04	17FEB04
10	Evaluate	Changes	10	1	.	.	18FEB04	02MAR04
11	Test Market	Changes	15	.	.	.	25FEB04	16MAR04
12	Changes	Production	5	1	1	.	17MAR04	23MAR04
13	Production		0	.	.	.	24MAR04	24MAR04
14	Marketing		0	.	.	.	25FEB04	25FEB04

Obs	E_START	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	01DEC03	05DEC03	01DEC03	05DEC03	0		
2	08DEC03	19DEC03	08DEC03	19DEC03	0		
3	08DEC03	12DEC03	21JAN04	27JAN04	40	prodeng	
4	08DEC03	12DEC03	15DEC03	19DEC03	10	deseng	
5	22DEC03	13JAN04	22DEC03	13JAN04	0		
6	15DEC03	29DEC03	28JAN04	10FEB04	0		
7	14JAN04	27JAN04	14JAN04	27JAN04	0		
8	14JAN04	27JAN04	14JAN04	27JAN04	0		
9	28JAN04	10FEB04	28JAN04	10FEB04	0		
10	11FEB04	24FEB04	18FEB04	02MAR04	0		
11	11FEB04	02MAR04	11FEB04	02MAR04	0		
12	03MAR04	09MAR04	03MAR04	09MAR04	0		
13	10MAR04	10MAR04	10MAR04	10MAR04	0		
14	11FEB04	11FEB04	10MAR04	10MAR04	0		

First, PROC CPM is invoked without reference to the RESID variable. The procedure ignores observations 2 and 3 in the RESOURCEIN data set (a message is written to the log), and the project is scheduled using the available resources; the supplementary level is not used because the project can be scheduled using only the primary resources by delaying it a few days. The project completion time is March 24, 2004 (see the schedule displayed in Output 4.20.3). The following program shows the invocation of PROC CPM.

```
proc cpm date='01dec03'd
         interval=weekday collapse
         data=widgr20 resin=resin20 holidata=holdata
         out=widgo20 resout=widgro20;
   activity task;
   duration days;
   successor succ;
   holiday hol;
   resource deseng prodeng engpool / period=per
                                     obstype=otype
                                     delayanalysis
                                     rcs avl;
   run;
```

Next, PROC CPM is invoked with the RESID= option, and the resulting Schedule data set is displayed in Output 4.20.4. The new schedule shows that the project completion time (10MAR04) has been reduced by two weeks as a result of using alternate resources.

Output 4.20.4 Alternate Resources Used

Scheduling with Alternate Resources
Alternate Resources Reduce Project Completion Time

Obs	task	succ	days	deseng	prodeng	engpool	Udeseng	Uprodeng	Uengpool	S_START
1	Approve Plan	Drawings	5	1	1	.	1	1	.	01DEC03
2	Drawings	Prototype	10	1	1	.	1	1	.	08DEC03
3	Study Market	Mkt. Strat.	5	.	1	.	.	.	1	08DEC03
4	Write Specs	Prototype	5	1	1	.	.	.	2	08DEC03
5	Prototype	Materials	15	1	1	1	1	1	1	22DEC03
6	Mkt. Strat.	Test Market	10	.	.	.	.	.	.	15DEC03
7	Materials	Init. Prod.	10	.	.	.	.	.	.	14JAN04
8	Facility	Init. Prod.	10	.	1	2	.	1	2	14JAN04
9	Init. Prod.	Test Market	10	.	.	2	.	.	2	28JAN04
10	Evaluate	Changes	10	1	.	.	1	.	.	11FEB04
11	Test Market	Changes	15	.	.	.	.	.	.	11FEB04
12	Changes	Production	5	1	1	.	1	1	.	03MAR04
13	Production		0	.	.	.	.	.	.	10MAR04
14	Marketing		0	.	.	.	.	.	.	11FEB04

Obs	S_FINISH	E_START	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	05DEC03	01DEC03	05DEC03	01DEC03	05DEC03	0		
2	19DEC03	08DEC03	19DEC03	08DEC03	19DEC03	0		
3	12DEC03	08DEC03	12DEC03	21JAN04	27JAN04	0		
4	12DEC03	08DEC03	12DEC03	15DEC03	19DEC03	0		
5	13JAN04	22DEC03	13JAN04	22DEC03	13JAN04	0		
6	29DEC03	15DEC03	29DEC03	28JAN04	10FEB04	0		
7	27JAN04	14JAN04	27JAN04	14JAN04	27JAN04	0		
8	27JAN04	14JAN04	27JAN04	14JAN04	27JAN04	0		
9	10FEB04	28JAN04	10FEB04	28JAN04	10FEB04	0		
10	24FEB04	11FEB04	24FEB04	18FEB04	02MAR04	0		
11	02MAR04	11FEB04	02MAR04	11FEB04	02MAR04	0		
12	09MAR04	03MAR04	09MAR04	03MAR04	09MAR04	0		
13	10MAR04	10MAR04	10MAR04	10MAR04	10MAR04	0		
14	11FEB04	11FEB04	11FEB04	10MAR04	10MAR04	0		

When resource substitution is allowed, the procedure adds a new variable prefixed by a 'U' for each resource variable; this new variable specifies the actual resources *used* for each activity (as opposed to the resource *required*). The activity 'Study Market' requires one production engineer who is tied up with the activity 'Drawings' on the 8th of December. Since resource substitution is allowed, the procedure uses an engineer from engpool as indicated by a missing value for Uprodeng and a '1' for Uengpool in the third observation. Likewise, the activity 'Write Specs' is scheduled by substituting one engineer from engpool for a design engineer and one for a production engineer to obtain Udeseng='.', Uprodeng='.', and Uengpool=2 in observation number 4. It is evident from the project finish date (S_FINISH=L_FINISH='10MAR04') that resource substitution has enabled the project to be completed without any delay. In fact, the DELAYANALYSIS variables indicate that there is no delay in any of the activities (R_DELAY=0 and DELAY_R=' ' for all activities). Note also that supplementary levels are not used (SUPPL_R=' ') for any of the resources (recall that use of supplementary levels is triggered by the specification of a finite value for DELAY).

The following program produced Output 4.20.4:

```
proc cpm date='01dec03'd
         interval=weekday collapse
         data=widgr20 resin=resin20 holidata=holdata
         out=widgalt resout=widralt;
   activity task;
   duration days;
   successor succ;
   holiday hol;
   resource deseng prodeng engpool / period=per
                                     obstype=otype
                                     delayanalysis
                                     resid=resid
                                     rcs avl;
   run;
```

The next two invocations of PROC CPM illustrate the use of both supplementary as well as alternate resources. Note from the output data set, displayed in Output 4.20.5, that once again the project is completed without any delay. Note also that the activity 'Write Specs' has used a supplementary resource whereas 'Study Market' has used an alternate resource. By default, when the DELAY= option is used, it forces the procedure to use supplementary resources before alternate resources. To invert this order so that alternate resources are used before supplementary resources, use the ALTBEFORESUP option in the RESOURCE statement, as illustrated in the second invocation of CPM in the following code. The resulting schedule is displayed in Output 4.20.6; this schedule is, in fact, the same as the schedule displayed in Output 4.20.4, obtained without the DELAY=0 and the ALTBEFORESUP options.

```
/* Invoke CPM with the DELAY=0 option */
proc cpm date='01dec03'd
         interval=weekday collapse
         data=widgr20 resin=resin20 holidata=holdata
         out=widgdsup resout=widrdsup;
   activity task;
   duration days;
   successor succ;
   holiday hol;
   resource deseng prodeng engpool / period=per
                                     obstype=otype
                                     delayanalysis
                                     delay=0
                                     resid=resid
                                     rcs avl;
   run;


/* Invoke CPM with the DELAY=0 and ALTBEFORESUP options */
proc cpm date='01dec03'd
         interval=weekday collapse
         data=widgr20 resin=resin20 holidata=holdata
         out=widgdsup resout=widrdsup;
   activity task;
   duration days;
   successor succ;
   holiday hol;
   resource deseng prodeng engpool / period=per
                                     obstype=otype
                                     delayanalysis
                                     delay=0
                                     resid=resid altbeforesup
                                     rcs avl;
   run;
```

Output 4.20.5 Supplementary Resources Used Before Alternate Resources

Scheduling with Alternate Resources
DELAY=0, Supplementary Resources Used instead of Alternate

Obs	task	succ	days	deseng	prodeng	engpool	Udeseng	Uprodeng	Uengpool	S_START
1	Approve Plan	Drawings	5	1	1	.	1	1	.	01DEC03
2	Drawings	Prototype	10	1	1	.	1	1	.	08DEC03
3	Study Market	Mkt. Strat.	5	.	1	.	.	.	1	08DEC03
4	Write Specs	Prototype	5	1	1	.	.	.	2	08DEC03
5	Prototype	Materials	15	1	1	1	1	1	1	22DEC03
6	Mkt. Strat.	Test Market	10	.	.	.	.	.	.	15DEC03
7	Materials	Init. Prod.	10	.	.	.	.	.	.	14JAN04
8	Facility	Init. Prod.	10	.	1	2	.	1	2	14JAN04
9	Init. Prod.	Test Market	10	.	.	2	.	.	2	28JAN04
10	Evaluate	Changes	10	1	.	.	1	.	.	11FEB04
11	Test Market	Changes	15	.	.	.	.	.	.	11FEB04
12	Changes	Production	5	1	1	.	1	1	.	03MAR04
13	Production		0	.	.	.	.	.	.	10MAR04
14	Marketing		0	.	.	.	.	.	.	11FEB04

Obs	S_FINISH	E_START	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	05DEC03	01DEC03	05DEC03	01DEC03	05DEC03	0		
2	19DEC03	08DEC03	19DEC03	08DEC03	19DEC03	0		
3	12DEC03	08DEC03	12DEC03	21JAN04	27JAN04	0		
4	12DEC03	08DEC03	12DEC03	15DEC03	19DEC03	0		
5	13JAN04	22DEC03	13JAN04	22DEC03	13JAN04	0		
6	29DEC03	15DEC03	29DEC03	28JAN04	10FEB04	0		
7	27JAN04	14JAN04	27JAN04	14JAN04	27JAN04	0		
8	27JAN04	14JAN04	27JAN04	14JAN04	27JAN04	0		
9	10FEB04	28JAN04	10FEB04	28JAN04	10FEB04	0		
10	24FEB04	11FEB04	24FEB04	18FEB04	02MAR04	0		
11	02MAR04	11FEB04	02MAR04	11FEB04	02MAR04	0		
12	09MAR04	03MAR04	09MAR04	03MAR04	09MAR04	0		
13	10MAR04	10MAR04	10MAR04	10MAR04	10MAR04	0		
14	11FEB04	11FEB04	11FEB04	10MAR04	10MAR04	0		

Output 4.20.6 Alternate Resources Used Before Supplementary Resources

Scheduling with Alternate Resources
DELAY=0, Alternate Resources Used instead of Supplementary

Obs	task	succ	days	deseng	prodeng	engpool	Udeseng	Uprodeng	Uengpool	S_START
1	Approve Plan	Drawings	5	1	1	.	1	1	.	01DEC03
2	Drawings	Prototype	10	1	1	.	1	1	.	08DEC03
3	Study Market	Mkt. Strat.	5	.	1	.	.	.	1	08DEC03
4	Write Specs	Prototype	5	1	1	.	.	.	2	08DEC03
5	Prototype	Materials	15	1	1	1	1	1	1	22DEC03
6	Mkt. Strat.	Test Market	10	.	.	.	.	.	.	15DEC03
7	Materials	Init. Prod.	10	.	.	.	.	.	.	14JAN04
8	Facility	Init. Prod.	10	.	1	2	.	1	2	14JAN04
9	Init. Prod.	Test Market	10	.	.	2	.	.	2	28JAN04
10	Evaluate	Changes	10	1	.	.	1	.	.	11FEB04
11	Test Market	Changes	15	.	.	.	.	.	.	11FEB04
12	Changes	Production	5	1	1	.	1	1	.	03MAR04
13	Production		0	.	.	.	.	.	.	10MAR04
14	Marketing		0	.	.	.	.	.	.	11FEB04

Obs	S_FINISH	E_START	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	05DEC03	01DEC03	05DEC03	01DEC03	05DEC03	0		
2	19DEC03	08DEC03	19DEC03	08DEC03	19DEC03	0		
3	12DEC03	08DEC03	12DEC03	21JAN04	27JAN04	0		
4	12DEC03	08DEC03	12DEC03	15DEC03	19DEC03	0		
5	13JAN04	22DEC03	13JAN04	22DEC03	13JAN04	0		
6	29DEC03	15DEC03	29DEC03	28JAN04	10FEB04	0		
7	27JAN04	14JAN04	27JAN04	14JAN04	27JAN04	0		
8	27JAN04	14JAN04	27JAN04	14JAN04	27JAN04	0		
9	10FEB04	28JAN04	10FEB04	28JAN04	10FEB04	0		
10	24FEB04	11FEB04	24FEB04	18FEB04	02MAR04	0		
11	02MAR04	11FEB04	02MAR04	11FEB04	02MAR04	0		
12	09MAR04	03MAR04	09MAR04	03MAR04	09MAR04	0		
13	10MAR04	10MAR04	10MAR04	10MAR04	10MAR04	0		
14	11FEB04	11FEB04	11FEB04	10MAR04	10MAR04	0		

Example 4.21: PERT Assumptions and Calculations

This example illustrates the PERT statistical approach. Throughout this chapter, it has been assumed that the activity duration times are precise values determined uniquely. In practice, however, each activity is subject to a number of chance sources of variation and it is impossible to know, a priori, the duration of the activity. The PERT statistical approach is used to include uncertainty about durations in scheduling. For a detailed discussion about various assumptions, techniques, and cautions related to the PERT approach, refer to Moder, Phillips, and Davis (1983) and Elmaghraby (1977). A simple model is used here to illustrate how PROC CPM can incorporate some of these ideas. A more detailed example can be found in *SAS/OR Software: Project Management Examples*.

Consider the widget manufacturing example. To perform PERT analysis, you need to provide three estimates of activity duration: a pessimistic estimate (tp), an optimistic estimate (to), and a modal estimate (tm). These three estimates are used to obtain a weighted average that is assumed to be a reasonable estimate of the activity duration. The time estimates for the activities must be independent for the analysis to be considered valid. Furthermore, the distribution of activity duration times is purely hypothetical, as no statistical sampling is likely to be feasible on projects of a unique nature to be accomplished at some indeterminate time in the future. Often, the time estimates used are based on past experience with similar projects.

To derive the formula for the mean, you must assume some functional form for the unknown distribution. The well-known Beta distribution is commonly used, as it has the desirable properties of being contained inside a finite interval and can be symmetric or skewed, depending on the location of the mode relative to the optimistic and pessimistic estimates. A linear approximation of the exact formula for the mean of the beta distribution weights the three time estimates as follows:

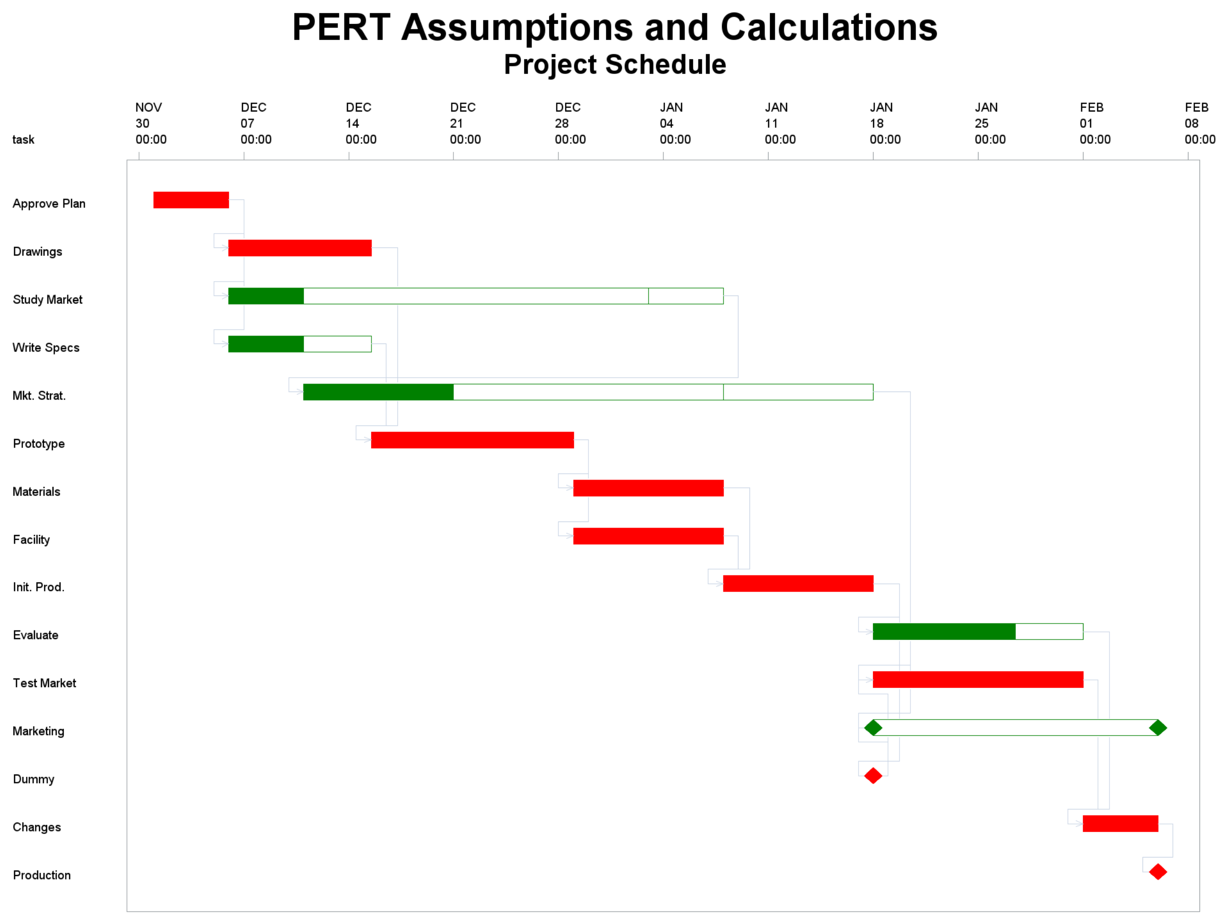
```
(tp + (4*tm) + to) / 6
```

The following program saves the network (AOA format) from Example 4.2 with three estimates of activity durations in a SAS data set. The DATA step also calculates the weighted average duration for each activity. Following the DATA step, PROC CPM is invoked to produce the schedule plotted on a Gantt chart in Output 4.21.1. The E_FINISH time for the final activity in the project contains the mean project completion time based on the duration estimates that are used.

```
title 'PERT Assumptions and Calculations';
 /* Activity-on-Arc representation of the project
    with three duration estimates */
data widgpert;
   format task $12. ;
   input task & tail head tm tp to;
   dur = (tp + 4*tm + to) / 6;
   datalines;
Approve Plan    1   2    5   7   3
Drawings        2   3   10  11   6
Study Market    2   4    5   7   3
Write Specs     2   3    5   7   3
Prototype       3   5   15  12   9
Mkt. Strat.     4   6   10  11   9
Materials       5   7   10  12   8
Facility        5   7   10  11   9
Init. Prod.     7   8   10  12   8
Evaluate        8   9    9  13   8
Test Market     6   9   14  15  13
Changes         9  10    5   6   4
Production     10  11    0   0   0
Marketing       6  12    0   0   0
Dummy           8   6    0   0   0
;

proc cpm data=widgpert out=sched
     date='1dec03'd;
   tailnode tail;
   headnode head;
   duration dur;
   id task;
   run;

proc sort;
   by e_start;
   run;
```

Some words of caution are worth mentioning with regard to the traditional PERT approach. The estimate of the mean project duration obtained in this instance always underestimates the true value since the length of a critical path is a convex function of the activity durations. The original PERT model developed by Malcolm et al. (1959) provides a way to estimate the variance of the project duration as well as calculating the probabilities of meeting certain target dates and so forth. Their analysis relies on an implicit assumption that you may ignore all activities that are not on the critical path in the deterministic problem that is derived by setting the activity durations equal to the mean value of their distributions. It then applies the Central Limit Theorem to the duration of this critical path and interprets the result as pertaining to the project duration.

Output 4.21.1 PERT Statistical Estimates: Gantt Chart



However, when the activity durations are random variables, each path of the project network is a likely candidate to be the critical path. Every outcome of the activity durations could result in a different longest path. Furthermore, there could be several dependent paths in the network in the sense that they share at least one common arc. Thus, in the most general case, the length of a longest path would be the maximum of a set of, possibly dependent, random variables. Evaluating or approximating the distribution of the longest path, even under very specific distributional assumptions on the activity durations is not a very easy problem. It is not surprising that this topic is the subject of much research.

In view of the inaccuracies that can stem from the original PERT assumptions, many people prefer to resort to the use of Monte Carlo Simulation. Van Slyke (1963) made the first attempt at straightforward simulation to analyze the distribution of the critical path. Refer to Elmaghraby (1977) for a detailed synopsis of the pitfalls of making traditional PERT assumptions and for an introduction to simulation techniques for activity networks.

Example 4.22: Scheduling Course - Teacher Combinations

This example demonstrates the use of PROC CPM for a typical scheduling problem that may not necessarily fit into a conventional project management scenario. Such problems abound in practice and can usually be solved using a mathematical programming model. Here, the problem is modeled as a resource-allocation problem using PROC CPM, illustrating the richness of the modeling environment that is available with the SAS System. (Refer also to Kulkarni (1991) and *SAS/OR Software: Project Management Examples* for another example of course scheduling using PROC CPM.)

A committee for academically gifted children wishes to conduct some special classes on weekends. There are four subjects that are to be taught and a number of teachers available to teach them. Only certain course-teacher combinations are allowed. There is a constraint on the number of rooms that are available and some teachers may not be able to teach at certain times. Possible class times are one-hour periods between 9 a.m. and 12 noon on Saturdays and Sundays. The goal is to determine a feasible schedule of classes specifying the teacher that is to teach each class.

Suppose that there are four courses, c1, c2, c3, and c4, and three teachers, t1, t2, and t3. There are several ways of modeling this problem; one possible way is to form distinct classes for each possible course-teacher combination and treat each of these as a distinct activity that needs to be scheduled. For example, if course c1 can be taught by teachers t1, t2, and t3, define three activities, 'c1t1', 'c1t2', and 'c1t3'. The resources for this problem are the courses, the teachers, and the number of rooms. In particular, the resources needed for a particular activity, say, 'c1t3', are c1 and t3.

The following constraints are imposed:

- Course 1 can be taught by Teachers 1, 2, and 3; Course 2 can be taught by Teachers 1 and 3; Course 3 can be taught by Teachers 1, 2, and 3; and Course 4 can be taught by Teachers 1 and 2.
- The total number of classes taught at any time cannot exceed NROOMS.
- Class 'citj' (if such a course-teacher combination is allowed) can be taught only at times when teacher tj is available.
- At any given time, a teacher can teach only one class.
- At any given time, only one class is to be taught for any given course.

The following program uses PROC CPM to schedule the classes. The schedule is obtained in terms of unformatted numeric values; the times 1, 2, 3, 4, 5, and 6 are interpreted as the six different time slots that are possible, namely, Saturday 9, 10, and 11 a.m. and Sunday 9, 10, and 11 a.m.

The data set CLASSES is the Activity data set, and it indicates the possible course-teacher combinations and identifies the specific room, teacher, and course as the resources required. For each activity, the duration is 1 unit. Note that, in this example, there are no precedence constraints between the activities; the resource availability dictates the schedule entirely. However, there may be situations (such as prerequisite courses) that impose precedence constraints.

The Resource data set, RESOURCE, specifies resource availabilities. The period variable, per, indicates the time period from which resources are available. Since only one class corresponding to a given course is to be taught at a given time, the availability for c1 – c4 is specified as '1'. Teacher 2 is available only on Sunday;

thus, specify the availability of t2 to be 1 from time period 4. The total number of rooms available at a given time is three. Thus, no more than three classes can be scheduled at a given time.

In the invocation of PROC CPM, the STOPDATE= option is used in the RESOURCE statement, thus restricting resource constrained scheduling to the first six time periods. Not all of the specified activities may be scheduled within the time available, in which case the unscheduled activities represent course-teacher combinations that are not feasible under the given conditions. The schedule obtained by PROC CPM is saved in a data set that is displayed, in Output 4.22.1, after formatting the activity names and the schedule times appropriately. Note that, in this example, all the course-teacher combinations are scheduled within the two-day time period.

```
title 'Scheduling Course / Teacher Combinations';
data classes;
   input class $  succ $ dur  c1-c4  t1-t3  nrooms;
   datalines;
c1t1  .   1   1  .  .  .  1  .  .   1
c1t2  .   1   1  .  .  .  .  1  .   1
c1t3  .   1   1  .  .  .  .  .  1   1
c2t1  .   1   .  1  .  .  1  .  .   1
c2t3  .   1   .  1  .  .  .  .  1   1
c3t1  .   1   .  .  1  .  1  .  .   1
c3t2  .   1   .  .  1  .  .  1  .   1
c3t3  .   1   .  .  1  .  .  .  1   1
c4t1  .   1   .  .  .  1  1  .  .   1
c4t2  .   1   .  .  .  1  .  1  .   1
;

data resource;
   input per c1-c4  t1-t3  nrooms;
   datalines;
1       1   1  1  1  1  .  1   3
4       .   .  .  .  .  1  .   .
;



proc cpm data=classes out=sched
     resin=resource;
   activity   class;
   duration   dur;
   successor  succ;
   resource   c1-c4 t1-t3 nrooms / period=per stopdate=6;
   run;

proc format;
   value classtim
      1 = 'Saturday  9:00-10:00'
      2 = 'Saturday 10:00-11:00'
      3 = 'Saturday 11:00-12:00'
      4 = 'Sunday    9:00-10:00'
      5 = 'Sunday   10:00-11:00'
      6 = 'Sunday   11:00-12:00'
      7 = 'Not Scheduled'
      ;
   value $classt
```

```
      c1t1 = 'Class 1, Teacher 1'
      c1t2 = 'Class 1, Teacher 2'
      c1t3 = 'Class 1, Teacher 3'
      c2t1 = 'Class 2, Teacher 1'
      c2t2 = 'Class 2, Teacher 2'
      c2t3 = 'Class 2, Teacher 3'
      c3t1 = 'Class 3, Teacher 1'
      c3t2 = 'Class 3, Teacher 2'
      c3t3 = 'Class 3, Teacher 3'
      c4t1 = 'Class 4, Teacher 1'
      c4t2 = 'Class 4, Teacher 2'
      c4t3 = 'Class 4, Teacher 3'
      ;

data schedtim;
set sched;
format classtim classtim.;
format class    $classt.;
if (s_start <= 6) then classtim = s_start;
else                   classtim = 7;
run;

title2 'Schedule of Classes';
proc print;
   id class;
   var classtim;
   run;
```

Output 4.22.1 Class Schedule

Scheduling Course / Teacher Combinations
Schedule of Classes

class	classtim
Class 1, Teacher 1	Saturday 9:00-10:00
Class 1, Teacher 2	Sunday 9:00-10:00
Class 1, Teacher 3	Saturday 10:00-11:00
Class 2, Teacher 1	Saturday 10:00-11:00
Class 2, Teacher 3	Saturday 9:00-10:00
Class 3, Teacher 1	Saturday 11:00-12:00
Class 3, Teacher 2	Sunday 10:00-11:00
Class 3, Teacher 3	Sunday 9:00-10:00
Class 4, Teacher 1	Sunday 9:00-10:00
Class 4, Teacher 2	Sunday 11:00-12:00

There may be several other constraints that you want to impose on the courses scheduled. These can usually be modeled suitably by changing the resource availability profile. For example, suppose that you want to schedule more classes at 10 a.m. and fewer at other times. The following program creates a new Resource data set, RESOURC2, that changes the number of rooms available. Again, PROC CPM is invoked with the STOPDATE= option, and the resulting schedule is displayed in Output 4.22.2. The schedule can also be displayed graphically using the NETDRAW procedure, as illustrated in a similar problem in Example 9.16 in Chapter 9, "The NETDRAW Procedure."

```
data resourc2;
   input per c1-c4  t1-t3  nrooms;
   datalines;
1       1   1  1  1  1  .  1   1
2       .   .  .  .  .  .  .   3
3       .   .  .  .  .  .  .   2
4       .   .  .  .  .  1  .   1
5       .   .  .  .  .  .  .   3
;


proc cpm data=classes out=sched2
     resin=resourc2;
   activity   class;
   duration   dur;
   successor  succ;
   resource   c1-c4 t1-t3 nrooms / period=per stopdate=6;
   run;

data schedtim;
set sched2;
format classtim classtim.;
format class    $classt.;
if (s_start <= 6) then classtim = s_start;
else                   classtim = 7;
run;

title2 'Alternate Schedule with Additional Constraints';
proc print;
   id class;
   var classtim;
   run;
```

Output 4.22.2 Alternate Class Schedule

Scheduling Course / Teacher Combinations
Alternate Schedule with Additional Constraints

class	classtim
Class 1, Teacher 1	Saturday 9:00-10:00
Class 1, Teacher 2	Sunday 9:00-10:00
Class 1, Teacher 3	Saturday 10:00-11:00
Class 2, Teacher 1	Saturday 10:00-11:00
Class 2, Teacher 3	Saturday 11:00-12:00
Class 3, Teacher 1	Saturday 11:00-12:00
Class 3, Teacher 2	Sunday 10:00-11:00
Class 3, Teacher 3	Sunday 11:00-12:00
Class 4, Teacher 1	Sunday 10:00-11:00
Class 4, Teacher 2	Sunday 11:00-12:00

Example 4.23: Multiproject Scheduling

This example illustrates multiproject scheduling. Consider a Survey project that contains three phases, Plan, Prepare, and Implement, with each phase containing more than one activity. You can consider each phase of the project as a subproject within the master project, Survey. Each subproject in turn contains the lowest level activities, also referred to as the leaf tasks. The Activity data set, containing the task durations, project hierarchy, and the precedence constraints, is displayed in Output 4.23.1.

The PROJECT and ACTIVITY variables together define the project hierarchy using the parent/child relationship. Thus, the subproject, ‘Plan’, contains the two leaf tasks, ‘plan sur’ and ‘design q’. Precedence constraints are specified between leaf tasks as well as between subprojects. For example, the subproject ‘Prepare’ is followed by the subproject ‘Implement’. Durations are specified for all the tasks in the project, except for the master project ‘Survey’.

In addition to the Activity data set, define a Holiday data set, also displayed in Output 4.23.1.

Output 4.23.1 Survey Project

Survey Project
Activity Data Set SURVEY

Obs	id	activity	duration	succ1	succ2	succ3	project
1	Plan Survey	plan sur	4	hire per	design q		Plan
2	Hire Personnel	hire per	5	trn per			Prepare
3	Design Questionnaire	design q	3	trn per	select h	print q	Plan
4	Train Personnel	trn per	3				Prepare
5	Select Households	select h	3				Prepare
6	Print Questionnaire	print q	4				Prepare
7	Conduct Survey	cond sur	10	analyze			Implement
8	Analyze Results	analyze	6				Implement
9	Plan	Plan	6				Survey
10	Prepare	Prepare	8	Implement			Survey
11	Implement	Implement	18				Survey
12	Survey Project	Survey	.				

Survey Project
Holiday Data Set HOLIDATA

Obs	hol
1	09APR04

The following statements invoke PROC CPM with a PROJECT statement identifying the parent task for each subtask in the Survey project. The calendar followed is a weekday calendar with a holiday defined on April 9, 2004. The ORDERALL option on the PROJECT statement creates the ordering variables ES_ASC and LS_ASC in the Schedule data set, and the ADDWBS option creates a work breakdown structure code for the project. The Schedule data set is displayed in Output 4.23.2, after being sorted by the variable ES_ASC.

The PROJ_DUR variable is missing for all the leaf tasks, and it contains the project duration for the supertasks. The project duration is computed as the span of all the subtasks of the supertask. The PROJ_LEV variable specifies the level of the subtask within the tree defining the project hierarchy, starting with the level '0' for the master project (or the root), 'Survey'. The variable WBS_CODE contains the Work Breakdown Structure code defined by the CPM procedure using the project hierarchy.

```
proc cpm data=survey date='29mar04'd out=survout1
         interval=weekday holidata=holidata;
   activity   activity;
   successor  succ1-succ3;
   duration   duration;
   id         id;
   holiday    hol;
   project    project / orderall addwbs;
   run;

proc sort;
   by es_asc;
   run;

title 'Conducting a Market Survey';
title2 'Early and Late Start Schedule';
proc print;
   run;
```

Output 4.23.2 Survey Project Schedule

Conducting a Market Survey
Early and Late Start Schedule

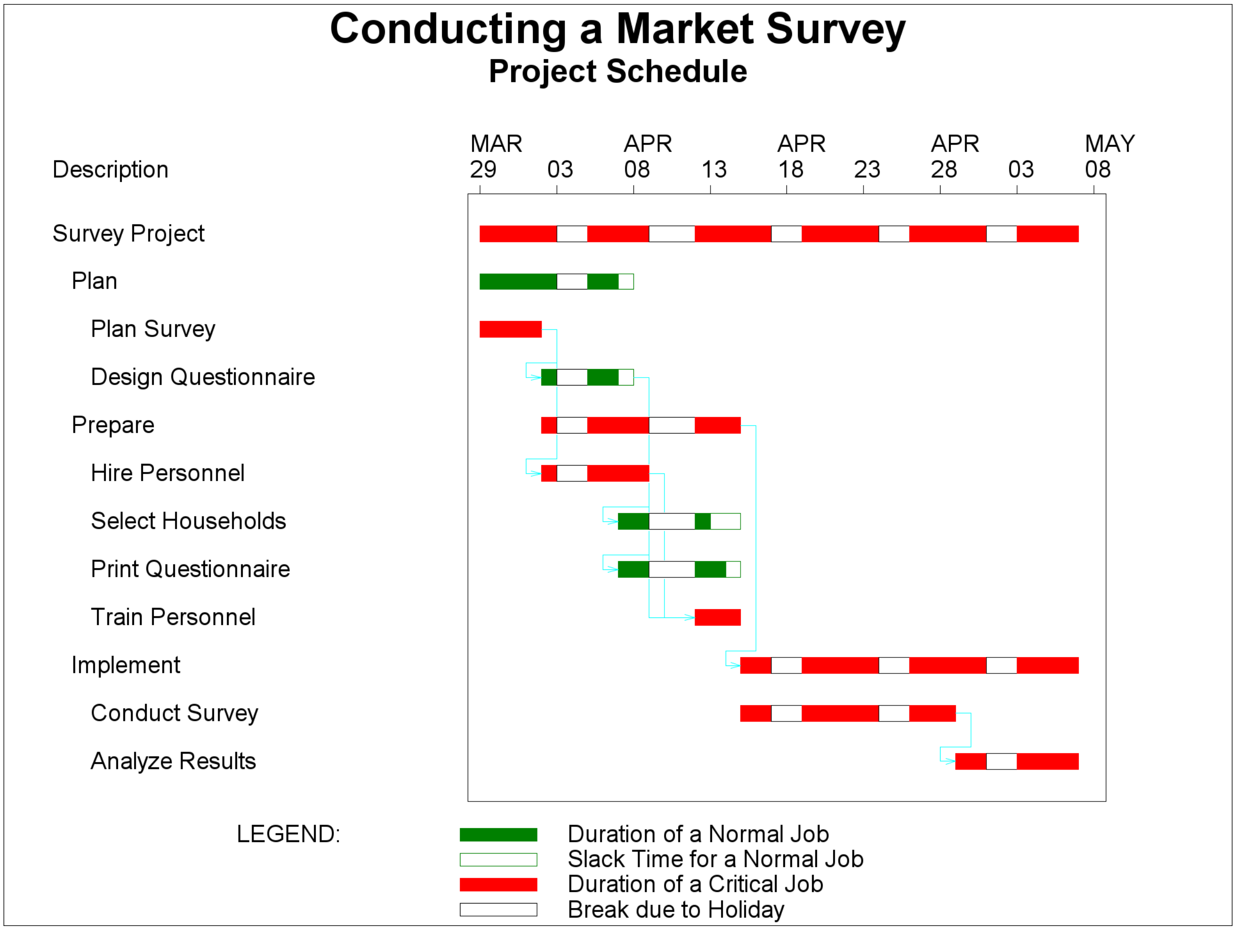
Obs	project	PROJ_DUR	PROJ_LEV	WBS_CODE	activity	succ1	succ2	succ3	duration
1		28	0	0	Survey				.
2	Survey	7	1	0.0	Plan				6
3	Plan	.	2	0.0.0	plan sur	hire per	design q		4
4	Plan	.	2	0.0.1	design q	trn per	select h	print q	3
5	Survey	8	1	0.1	Prepare	Implement			8
6	Prepare	.	2	0.1.0	hire per	trn per			5
7	Prepare	.	2	0.1.2	select h				3
8	Prepare	.	2	0.1.3	print q				4
9	Prepare	.	2	0.1.1	trn per				3
10	Survey	16	1	0.2	Implement				18
11	Implement	.	2	0.2.0	cond sur	analyze			10
12	Implement	.	2	0.2.1	analyze				6

Obs	id	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT	ES_ASC	LS_ASC
1	Survey Project	29MAR04	06MAY04	29MAR04	06MAY04	0	0	0	0
2	Plan	29MAR04	06APR04	29MAR04	07APR04	1	1	1	1
3	Plan Survey	29MAR04	01APR04	29MAR04	01APR04	0	0	2	2
4	Design Questionnaire	02APR04	06APR04	05APR04	07APR04	1	0	3	3
5	Prepare	02APR04	14APR04	02APR04	14APR04	0	0	4	4
6	Hire Personnel	02APR04	08APR04	02APR04	08APR04	0	0	5	5
7	Select Households	07APR04	12APR04	12APR04	14APR04	2	2	6	8
8	Print Questionnaire	07APR04	13APR04	08APR04	14APR04	1	1	7	6
9	Train Personnel	12APR04	14APR04	12APR04	14APR04	0	0	8	7
10	Implement	15APR04	06MAY04	15APR04	06MAY04	0	0	9	9
11	Conduct Survey	15APR04	28APR04	15APR04	28APR04	0	0	10	10
12	Analyze Results	29APR04	06MAY04	29APR04	06MAY04	0	0	11	11

Next, a Gantt chart of the master project schedule is produced with the subtasks of each project indented under the parent task. To produce the required indentation, prefix the Activity description (saved in the variable id) by a suitable number of blanks using a simple DATA step. The following program shows the DATA step and the invocation of the GANTT procedure; the resulting Gantt chart is plotted in Output 4.23.3. Note the precedence constraints between the two supertasks 'Prepare' and 'Implement'.

```
data gant;
length id $26.;
set survout1;
if proj_lev=1 then id="   "||id;
else if proj_lev=2 then id="      "||id;
run;

goptions hpos=80 vpos=43;
title c=black h=2 'Conducting a Market Survey';
title2 c=black h=1.5 'Project Schedule';

proc gantt graphics data=gant holidata=holidata;
   chart / holiday=(hol)
           interval=weekday
           skip=2 height=1.8
           nojobnum
           compress noextrange
           activity=activity succ=(succ1-succ3)
           cprec=cyan cmile=magenta
           caxis=black;
   id   id;
   run;
```

Output 4.23.3 Gantt Chart of Schedule



PROJ_LEV, WBS_CODE, and other project-related variables can be used to display selected information about specific subprojects, summary information about subprojects at a given level of the hierarchy, and more. For example, the following statements display the summary schedule of the first level subtasks of the Survey project (Output 4.23.4).

```
title 'Market Survey';
title2 'Summary Schedule';
proc print data=survout1;
   where proj_lev=1;
   id activity;
   var proj_dur duration e_start--t_float;
   run;
```

Output 4.23.4 Survey Project Summary

Market Survey
Summary Schedule

activity	PROJ_DUR	duration	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT
Plan	7	6	29MAR04	06APR04	29MAR04	07APR04	1
Prepare	8	8	02APR04	14APR04	02APR04	14APR04	0
Implement	16	18	15APR04	06MAY04	15APR04	06MAY04	0

The variable WBS_CODE in the Schedule data set (see Output 4.23.2) contains the Work Breakdown structure code defined by the CPM procedure. This code is defined to be '0.1' for the subproject 'Prepare'. Thus, the values of WBS_CODE for all subtasks of this subproject are prefixed by '0.1'. To produce reports for the subproject 'Prepare', you can use a simple WHERE clause to subset the required observations from the Schedule data set, as shown in the following statements.

```
title 'Market Survey';
title2 'Sub-Project Schedule';
proc print data=survout1;
   where substr(WBS_CODE,1,3) = "0.1";
   id activity;
   var project--activity duration e_start--t_float;
   run;
```

Output 4.23.5 Subproject Schedule

Market Survey
Sub-Project Schedule

activity	project	PROJ_DUR	PROJ_LEV	WBS_CODE	activity	duration	E_START	E_FINISH
Prepare	Survey	8	1	0.1	Prepare	8	02APR04	14APR04
hire per	Prepare	.	2	0.1.0	hire per	5	02APR04	08APR04
select h	Prepare	.	2	0.1.2	select h	3	07APR04	12APR04
print q	Prepare	.	2	0.1.3	print q	4	07APR04	13APR04
trn per	Prepare	.	2	0.1.1	trn per	3	12APR04	14APR04

activity	L_START	L_FINISH	T_FLOAT
Prepare	02APR04	14APR04	0
hire per	02APR04	08APR04	0
select h	12APR04	14APR04	2
print q	08APR04	14APR04	1
trn per	12APR04	14APR04	0

In the first invocation of PROC CPM, the Survey project is scheduled with only a specification for the project start date. Continuing, this example shows how you can impose additional constraints on the master project or on the individual subprojects.

First, suppose that you impose a FINISHBEFORE constraint on the Survey project by specifying the FBDATE to be May 10, 2004. The following program schedules the project with a *project start and finish* specification. The resulting summary schedule for the subprojects is shown in Output 4.23.6. The late finish time of the project is the 7th of May because there is a weekend on the 8th and 9th of May, 2004.

```
proc cpm data=survey date='29mar04'd out=survout2
         interval=weekday holidata=holidata
         fbdate='10may04'd; /* project finish date */
   activity   activity;
   successor  succ1-succ3;
   duration   duration;
   id         id;
   holiday    hol;
   project    project / orderall addwbs;
   run;

title 'Market Survey';
title2 'Summary Schedule: FBDATE Option';
proc print data=survout2;
   where proj_lev=1;  /* First level subprojects */
   id activity;
   var proj_dur duration e_start--t_float;
   run;
```

Output 4.23.6 Summary Schedule: FBDATE Option

Market Survey
Summary Schedule: FBDATE Option

activity	PROJ_DUR	duration	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT
Plan	7	6	29MAR04	06APR04	30MAR04	08APR04	2
Prepare	8	8	02APR04	14APR04	05APR04	15APR04	1
Implement	16	18	15APR04	06MAY04	16APR04	07MAY04	1

The procedure computes the backward pass of the schedule starting from the *project finish date*. Thus, the critical path is computed in the context of the entire project. If you want to obtain individual critical paths for each subproject, use the SEPCRIT option on the PROJECT statement. You can see the effect of this option in Output 4.23.7: all the subprojects have T_FLOAT = ‘0’.

```
proc cpm data=survey date='29mar04'd out=survout3
         interval=weekday holidata=holidata fbdate='10may04'd;
   activity   activity;
   successor  succ1-succ3;
   duration   duration;
   id         id;
   holiday    hol;
   project    project / orderall addwbs sepcrit;
   run;


title 'Market Survey';
```

```
title2 'Summary Schedule: FBDATE and SEPCRIT Options';
proc print data=survout3;
   where proj_lev=1;
   id activity;
   var proj_dur duration e_start--t_float;
run;
```

Output 4.23.7 Summary Schedule: FBDATE and SEPCRIT Options

Market Survey
Summary Schedule: FBDATE and SEPCRIT Options

activity	PROJ_DUR	duration	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT
Plan	7	6	29MAR04	06APR04	29MAR04	06APR04	0
Prepare	8	8	02APR04	14APR04	02APR04	14APR04	0
Implement	16	18	15APR04	06MAY04	15APR04	06MAY04	0

Now, suppose that, in addition to imposing a FINISHBEFORE constraint on the entire project, the project manager for each subproject specifies a desired duration for his or her subproject. In the present example, the variable duration has values '6', '8', and '18' for the three subprojects. By default these values are not used in either the backward or forward pass, even though they may represent desired durations for the corresponding subprojects. You can specify the USEPROJDUR option on the PROJECT statement to indicate that the procedure should use these specified durations to determine the late finish schedule for each of the subprojects. In other words, if the USEPROJDUR option is specified, the late finish for each subproject is constrained to be less than or equal to

$$\text{E\_START} + \text{duration}$$

and this value is used during the backward pass.

The summary schedule resulting from the use of the USEPROJDUR option is shown in Output 4.23.8. Note the difference in the schedules in Output 4.23.7 and Output 4.23.8. In Output 4.23.7, the *computed project duration*, PROJ_DUR, is used to set an upper bound on the late finish time of each subproject, while in Output 4.23.8, the *specified project duration* is used for the same purpose. Here, only the summary schedules are displayed; the effect of the two options on the subtasks within each subproject can be seen by displaying the entire schedule in each case. A Gantt chart of the entire project is displayed in Output 4.23.9.

```
proc cpm data=survey date='29mar04'd out=survout4
         interval=weekday holidata=holidata fbdate='10may04'd;
   activity   activity;
   successor  succ1-succ3;
   duration   duration;
   id         id;
   holiday    hol;
   project    project / orderall addwbs useprojdur;
   run;
```

```
title 'Market Survey';
title2 'Summary Schedule: FBDATE and USEPROJDUR Options';
```

```
proc print data=survout4;
   where proj_lev=1;
   id activity;
   var proj_dur duration e_start--t_float;
run;
```

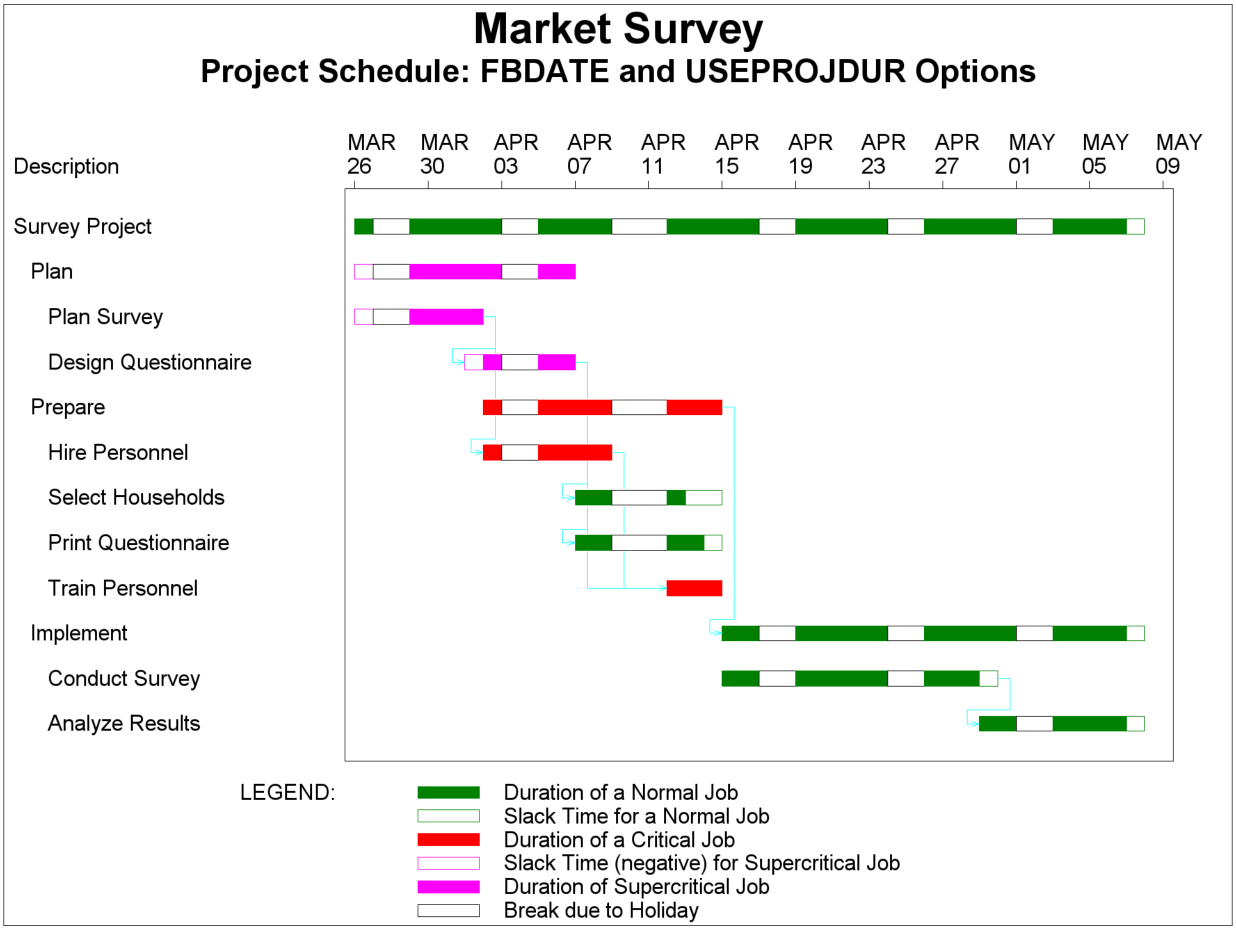
Output 4.23.8 Summary Schedule: FBDATE and USEPROJDUR Options

Market Survey
Summary Schedule: FBDATE and USEPROJDUR Options

activity	PROJ_DUR	duration	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT
Plan	7	6	29MAR04	06APR04	26MAR04	05APR04	-1
Prepare	8	8	02APR04	14APR04	02APR04	14APR04	0
Implement	16	18	15APR04	06MAY04	16APR04	07MAY04	1

```
data gant4;
   length id $26.;
   set survout4;
   if proj_lev=1 then id="   "||id;
   else if proj_lev=2 then id="      "||id;
   run;


proc sort;
   by es_asc;
   run;

goptions hpos=80 vpos=43;
title h=2 'Market Survey';
title2 h=1.5 'Project Schedule: FBDATE and USEPROJDUR Options';
proc gantt graphics data=gant4 holidata=holidata;
   chart / holiday=(hol)
           interval=weekday
           skip=2 scale=1.5 height=1.8
           nojobnum
           compress noextrange
           activity=activity succ=(succ1-succ3)
           cprec=cyan cmile=magenta
           caxis=black
           ;
   id   id;
   run;
```

Output 4.23.9 Gantt Chart of Schedule



The project schedule is further affected by the presence of any alignment dates on the individual activities or subprojects. For example, if the implementation phase of the project has a deadline of May 5, 2004, you can specify an alignment date and type variable with the appropriate values for the subproject 'Implement', as follows, and invoke PROC CPM with the ALIGNDATE and ALIGNTYPE statements, to obtain the new schedule, displayed in Output 4.23.10.

```
data survey2;
   format aldate date7.;
   set survey;
   if activity="Implement" then do;
      altype="fle";
      aldate='5may04'd;
   end;
   run;
```

```
proc cpm data=survey2 date='29mar04'd out=survout5
         interval=weekday holidata=holidata
         fbdate='10jun04'd;
   activity   activity;
   successor  succ1-succ3;
   duration   duration;
   id         id;
   holiday    hol;
   project    project / orderall addwbs sepcrit useprojdur;
   aligntype  altype;
   aligndate  aldate;
   run;

title 'Market Survey';
title2 'USEPROJDUR option and Alignment date';
proc print;
   where proj_lev=1;
   id activity;
   var proj_dur duration e_start--t_float;
   run;
```

Output 4.23.10 USEPROJDUR option and Alignment Date

Market Survey
USEPROJDUR option and Alignment date

activity	PROJ_DUR	duration	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT
Plan	7	6	29MAR04	06APR04	26MAR04	05APR04	-1
Prepare	8	8	02APR04	14APR04	01APR04	13APR04	-1
Implement	16	18	15APR04	06MAY04	14APR04	05MAY04	-1

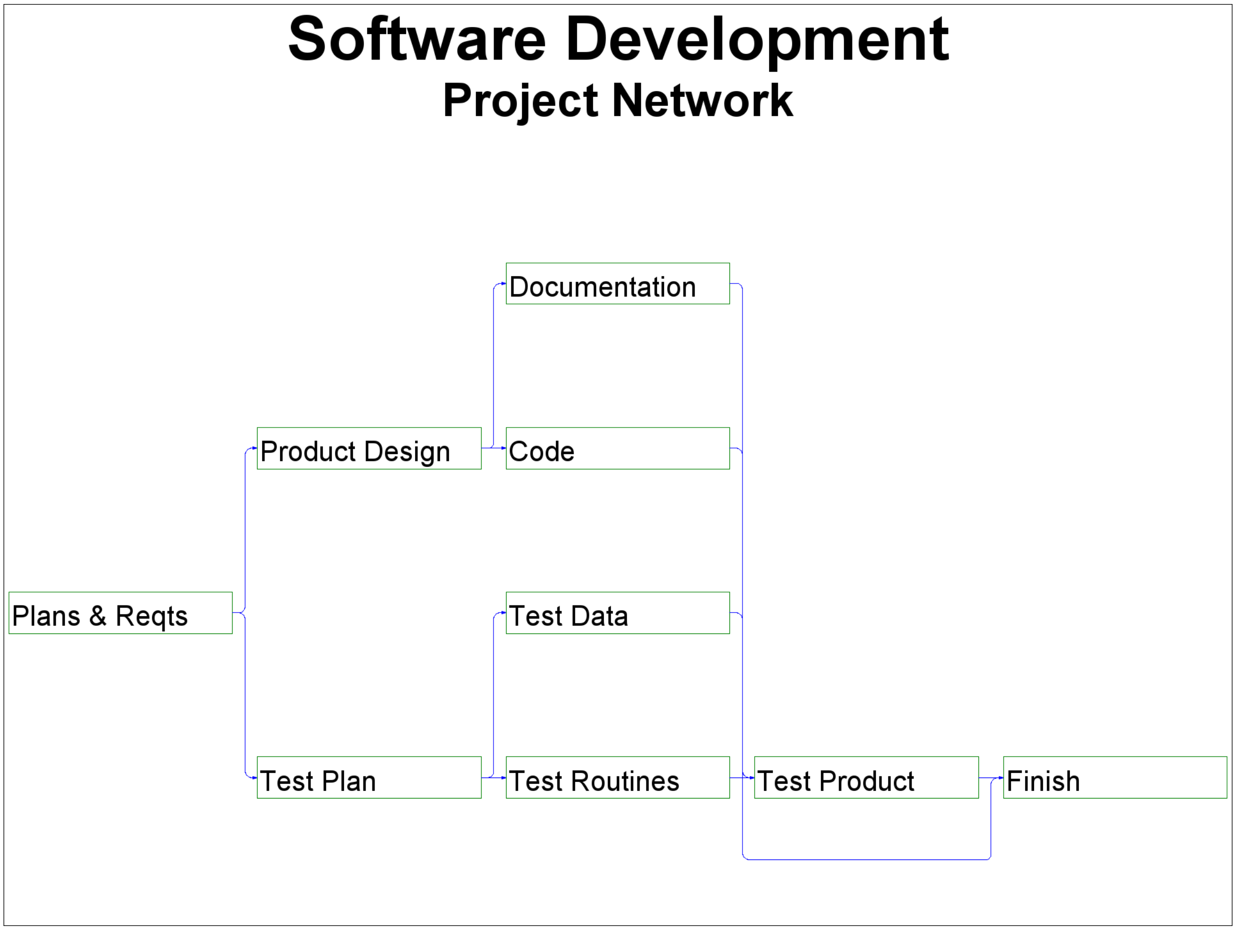
Example 4.24: Resource-Driven Durations and Resource Calendars

This example illustrates the effect of resource-driven durations and resource calendars on the schedule of a project involving multiple resources.

In projects that use manpower as a resource, the same activity may require different amounts of work from different people. Also, the work schedules and vacations may differ for each individual person. All of these factors may cause the schedules for the different resources used by the activity to differ from each other.

Consider a software project requiring two resources: a programmer and a tester. A network diagram displaying the activities and their precedence relationships is shown in Output 4.24.1.

Output 4.24.1 Software Project Network



Some of the activities in this project have a fixed duration, requiring the same length of time from both resources; others require a different number of days from the programmer and the tester. Further, some activities require only a fraction of the resource; for example, 'Documentation' requires only 20 percent of the programmer's time for a total of two man-days. The activities in the project, their durations (if fixed) in days, the total work required (if resource-driven) in days, the precedence constraints, and the resource requirements are displayed in Output 4.24.2. There are two observations for some of the activities ('Product Design' and 'Documentation') which require different amounts of work from each resource.

Output 4.24.2 Project Data

Software Development
Activity Data Set SOFTWARE

Activity	act	s1	s2	dur	mandays	Programmer	Tester
Plans & Reqts	1	2	3	2	.	1.0	1.0
Product Design	2	4	5	.	3	1.0	.
Product Design	2	.	.	.	1	.	1.0
Test Plan	3	6	7	3	.	.	1.0
Documentation	4	9	.	.	2	0.2	.
Documentation	4	.	.	.	1	.	0.5
Code	5	8	.	10	.	0.8	.
Test Data	6	8	.	5	.	.	0.5
Test Routines	7	8	.	5	.	.	0.5
Test Product	8	9	.	6	.	0.5	1.0
Finish	9	.	.	0	.	.	.

The following statements invoke PROC CPM with a WORK= specification on the RESOURCE statement, which identifies (in number of man-days, in this case) the amount of work required from each resource used by an activity. If the WORK variable has a missing value, the activity in that observation is assumed to have a fixed duration. The project is scheduled to start on April 12, 2004, and the activities are assumed to follow a five-day work week. Unlike fixed-duration scheduling, each resource used by an activity could have a different schedule; an activity is assumed to be finished only when all of its resources have finished working on it.

```
proc cpm data=software out=sftout ressched=rsftout
         date='12apr04'd interval=weekday resout=rout;
   act act;
   succ s1 s2;
   dur dur;
   res Programmer Tester / work=mandays
                     rschedid=Activity;
   id Activity;
run;
```

The individual resource schedules, as well as each activity's combined schedule, are saved in a Resource Schedule data set, RSFTOUT, requested by the RESSCHED= option on the CPM statement. This output data set (displayed in Output 4.24.3) is very similar to the Schedule data set and contains the activity variable and all the relevant schedule variables (E_START, E_FINISH, L_START, and so forth).

Output 4.24.3 Resource Schedule Data Set

Software Development
Resource Schedule Data Set RSFTOUT

Activity	act	RESOURCE	DUR_TYPE	dur	mandays	R_RATE	E_START	E_FINISH	L_START	L_FINISH
Plans & Reqts	1			2	.	.	12APR04	13APR04	12APR04	13APR04
Plans & Reqts	1	Programmer	FIXED	2	.	1.0	12APR04	13APR04	12APR04	13APR04
Plans & Reqts	1	Tester	FIXED	2	.	1.0	12APR04	13APR04	12APR04	13APR04
Product Design	2			3	.	.	14APR04	16APR04	14APR04	16APR04
Product Design	2	Programmer	RDRIVEN	3	3	1.0	14APR04	16APR04	14APR04	16APR04
Product Design	2	Tester	RDRIVEN	1	1	1.0	14APR04	14APR04	16APR04	16APR04
Test Plan	3			3	.	.	14APR04	16APR04	21APR04	23APR04
Test Plan	3	Tester	FIXED	3	.	1.0	14APR04	16APR04	21APR04	23APR04
Documentation	4			10	.	.	19APR04	30APR04	27APR04	10MAY04
Documentation	4	Programmer	RDRIVEN	10	2	0.2	19APR04	30APR04	27APR04	10MAY04
Documentation	4	Tester	RDRIVEN	2	1	0.5	19APR04	20APR04	07MAY04	10MAY04
Code	5			10	.	.	19APR04	30APR04	19APR04	30APR04
Code	5	Programmer	FIXED	10	.	0.8	19APR04	30APR04	19APR04	30APR04
Test Data	6			5	.	.	19APR04	23APR04	26APR04	30APR04
Test Data	6	Tester	FIXED	5	.	0.5	19APR04	23APR04	26APR04	30APR04
Test Routines	7			5	.	.	19APR04	23APR04	26APR04	30APR04
Test Routines	7	Tester	FIXED	5	.	0.5	19APR04	23APR04	26APR04	30APR04
Test Product	8			6	.	.	03MAY04	10MAY04	03MAY04	10MAY04
Test Product	8	Programmer	FIXED	6	.	0.5	03MAY04	10MAY04	03MAY04	10MAY04
Test Product	8	Tester	FIXED	6	.	1.0	03MAY04	10MAY04	03MAY04	10MAY04
Finish	9			0	.	.	11MAY04	11MAY04	11MAY04	11MAY04

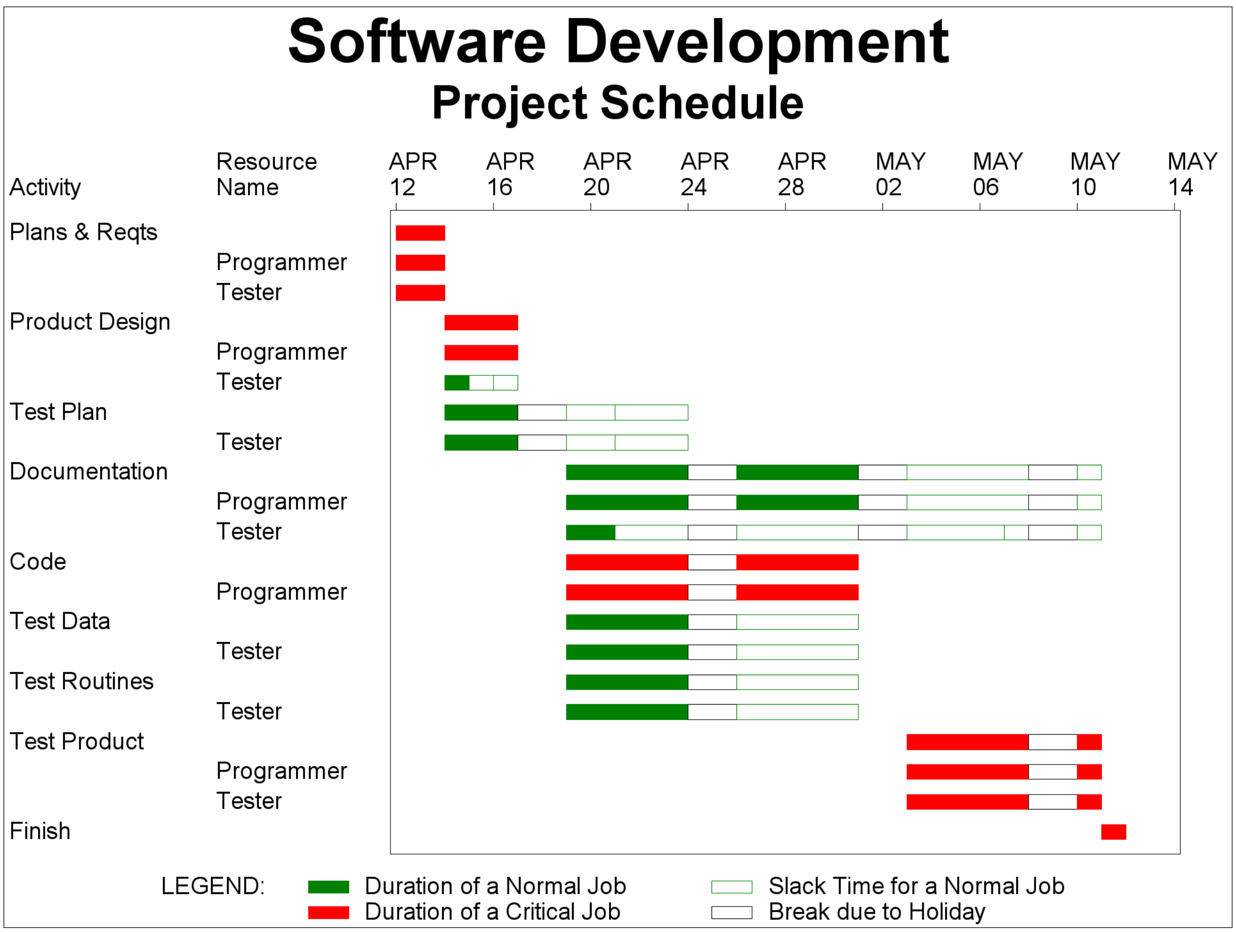
For each activity in the project, the Resource Schedule data set contains the schedule for the entire activity as well as the schedule for each resource used by the activity. The variable RESOURCE identifies the name of the resource to which the observation refers and has missing values for observations that refer to the entire activity's schedule. The value of the variable DUR_TYPE indicates whether the resource drives the activity's duration ('RDRIVEN') or not ('FIXED').

The DURATION variable, dur, indicates the duration of the activity for the resource identified in that observation. For resources that are of the driving type, the WORK variable, mandays, shows the total amount of work (in units of the INTERVAL parameter) required by the resource for the activity in that observation. The variable R_RATE shows the rate of usage of the resource for the relevant activity. For driving resources, the variable dur is computed as (mandays / R_RATE). Thus, for the Activity, 'Documentation', the programmer requires 10 days to complete 2 man-days of work at a rate of 20 percent per day, while the tester works at a rate of 50 percent requiring 2 days to complete 1 man-day of work.

```
pattern1 c=green v=s;      /* duration of a non-critical activity   */
pattern2 c=green v=e;      /* slack time for a noncrit. activity    */
pattern3 c=red   v=s;      /* duration of a critical activity       */
pattern4 c=magenta v=e;    /* slack time for a supercrit. activity  */
pattern5 c=magenta v=s;    /* duration of a supercrit. activity     */
pattern6 c=cyan v=s;       /* actual duration of an activity        */
pattern7 c=black v=e;      /* break due to a holiday                */
pattern8 c=blue v=s;       /* resource schedule of activity         */
pattern9 c=brown v=s;      /* baseline schedule of activity         */

title h=2 'Software Development';
title2 h=1.5 'Project Schedule';
```

A Gantt chart of the schedules for each resource is plotted in Output 4.24.4.

Output 4.24.4 Software Project Schedule



The daily utilization of the resources is also saved in a data set, ROUT, displayed in Output 4.24.5. The resource usage data set indicates that you need more than one tester on some days with both the early schedule (on the 14th, 19th, and 20th of April) and the late schedule (on the 7th and 10th of May).

Output 4.24.5 Resource Usage Data

Software Development
Resource Usage Data Set ROUT

Obs	_TIME_	EProgrammer	LProgrammer	ETester	LTester
1	12APR04	1.0	1.0	1.0	1.0
2	13APR04	1.0	1.0	1.0	1.0
3	14APR04	1.0	1.0	2.0	0.0
4	15APR04	1.0	1.0	1.0	0.0
5	16APR04	1.0	1.0	1.0	1.0
6	19APR04	1.0	0.8	1.5	0.0
7	20APR04	1.0	0.8	1.5	0.0
8	21APR04	1.0	0.8	1.0	1.0
9	22APR04	1.0	0.8	1.0	1.0
10	23APR04	1.0	0.8	1.0	1.0
11	26APR04	1.0	0.8	0.0	1.0
12	27APR04	1.0	1.0	0.0	1.0
13	28APR04	1.0	1.0	0.0	1.0
14	29APR04	1.0	1.0	0.0	1.0
15	30APR04	1.0	1.0	0.0	1.0
16	03MAY04	0.5	0.7	1.0	1.0
17	04MAY04	0.5	0.7	1.0	1.0
18	05MAY04	0.5	0.7	1.0	1.0
19	06MAY04	0.5	0.7	1.0	1.0
20	07MAY04	0.5	0.7	1.0	1.5
21	10MAY04	0.5	0.7	1.0	1.5
22	11MAY04	0.0	0.0	0.0	0.0

Suppose now that you have only one tester and one programmer. You can determine a resource-constrained schedule using PROC CPM (as in the fixed duration case) by specifying a resource availability data set, RESIN (Output 4.24.6).

Output 4.24.6 Resource Availability Data

Software Development
Resource Availability Data Set

Obs	per	otype	Programmer	Tester
1	12APR04	reslevel	1	1

The following statements invoke PROC CPM, and the resulting Resource Schedule data set is displayed in Output 4.24.7. The ADDCAL option on the RESOURCE statement creates a variable in the Resource Schedule data set which identifies the activity or resource calendar. The project still finishes on May 11, but some of the activities ('Test Plan', 'Documentation', 'Test Data', and 'Test Routines') are delayed. The resource-constrained schedule is plotted on a Gantt chart in Output 4.24.8; both resources follow the same weekday calendar.

```
proc cpm data=software resin=resin
         out=sftout1 resout=rout1
         rsched=rsftout1
         date='12apr04'd interval=weekday;
   act act;
   succ s1 s2;
   dur dur;
   res Programmer Tester / work=mandays addcal
                          obstype=otype
                          period=per
                          rschedid=Activity;
   id Activity;
run;
```

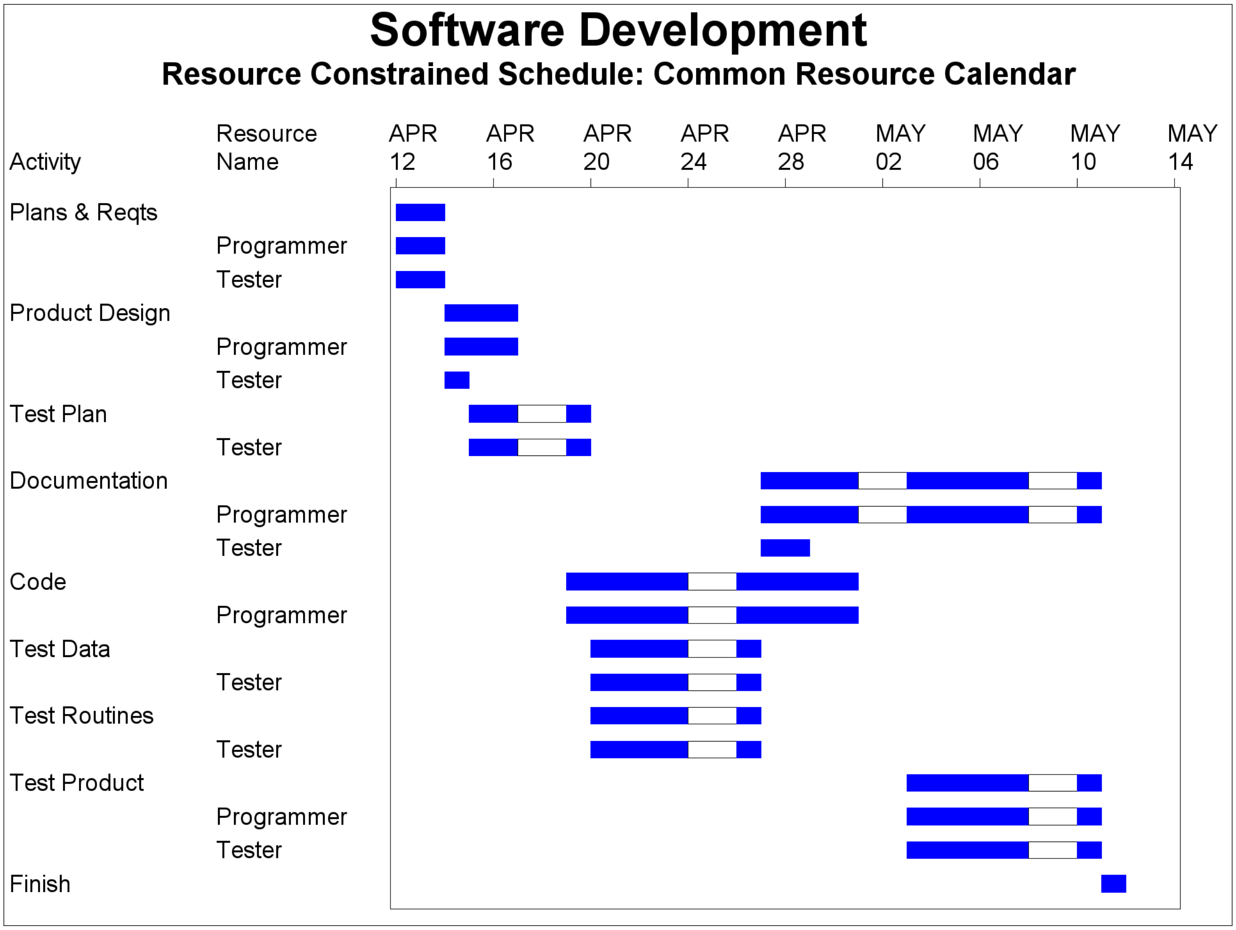
Output 4.24.7 Resource-Constrained Schedule: Common Calendar

Software Development
Resource Constrained Schedule: Common Resource Calendar

Activity	act	_CAL_	RESOURCE	DUR_TYPE	dur	mandays	R_RATE	S_START	S_FINISH
Plans & Reqts	1	0			2	.	.	12APR04	13APR04
Plans & Reqts	1	0	Programmer	FIXED	2	.	1.0	12APR04	13APR04
Plans & Reqts	1	0	Tester	FIXED	2	.	1.0	12APR04	13APR04
Product Design	2	0			3	.	.	14APR04	16APR04
Product Design	2	0	Programmer	RDRIVEN	3	3	1.0	14APR04	16APR04
Product Design	2	0	Tester	RDRIVEN	1	1	1.0	14APR04	14APR04
Test Plan	3	0			3	.	.	15APR04	19APR04
Test Plan	3	0	Tester	FIXED	3	.	1.0	15APR04	19APR04
Documentation	4	0			10	.	.	27APR04	10MAY04
Documentation	4	0	Programmer	RDRIVEN	10	2	0.2	27APR04	10MAY04
Documentation	4	0	Tester	RDRIVEN	2	1	0.5	27APR04	28APR04
Code	5	0			10	.	.	19APR04	30APR04
Code	5	0	Programmer	FIXED	10	.	0.8	19APR04	30APR04
Test Data	6	0			5	.	.	20APR04	26APR04
Test Data	6	0	Tester	FIXED	5	.	0.5	20APR04	26APR04
Test Routines	7	0			5	.	.	20APR04	26APR04
Test Routines	7	0	Tester	FIXED	5	.	0.5	20APR04	26APR04
Test Product	8	0			6	.	.	03MAY04	10MAY04
Test Product	8	0	Programmer	FIXED	6	.	0.5	03MAY04	10MAY04
Test Product	8	0	Tester	FIXED	6	.	1.0	03MAY04	10MAY04
Finish	9	0			0	.	.	11MAY04	11MAY04

Activity	E_START	E_FINISH	L_START	L_FINISH
Plans & Reqts	12APR04	13APR04	12APR04	13APR04
Plans & Reqts	12APR04	13APR04	12APR04	13APR04
Plans & Reqts	12APR04	13APR04	12APR04	13APR04
Product Design	14APR04	16APR04	14APR04	16APR04
Product Design	14APR04	16APR04	14APR04	16APR04
Product Design	14APR04	14APR04	16APR04	16APR04
Test Plan	14APR04	16APR04	21APR04	23APR04
Test Plan	14APR04	16APR04	21APR04	23APR04
Documentation	19APR04	30APR04	27APR04	10MAY04
Documentation	19APR04	30APR04	27APR04	10MAY04
Documentation	19APR04	20APR04	07MAY04	10MAY04
Code	19APR04	30APR04	19APR04	30APR04
Code	19APR04	30APR04	19APR04	30APR04
Test Data	19APR04	23APR04	26APR04	30APR04
Test Data	19APR04	23APR04	26APR04	30APR04
Test Routines	19APR04	23APR04	26APR04	30APR04
Test Routines	19APR04	23APR04	26APR04	30APR04
Test Product	03MAY04	10MAY04	03MAY04	10MAY04
Test Product	03MAY04	10MAY04	03MAY04	10MAY04
Test Product	03MAY04	10MAY04	03MAY04	10MAY04
Finish	11MAY04	11MAY04	11MAY04	11MAY04

Output 4.24.8 Resource-Constrained Schedule



Now suppose that the tester switches to part-time employment, working only four days a week. Thus, the two resources have different calendars. To determine the effect this change has on the project schedule, define a calendar data set identifying calendar '1' as having a holiday on Friday (see Output 4.24.9). In a new resource availability data set (also displayed in Output 4.24.9), associate calendar '1' with the resource Tester and calendar '0' with the resource Programmer. '0' refers to the default calendar, which is the weekday calendar for this project (since INTERVAL = WEEKDAY).

Output 4.24.9 Resource and Calendar Data

Software Development
Calendar Data Set CALENDAR

Obs	_cal_	_fri_
1	1	holiday

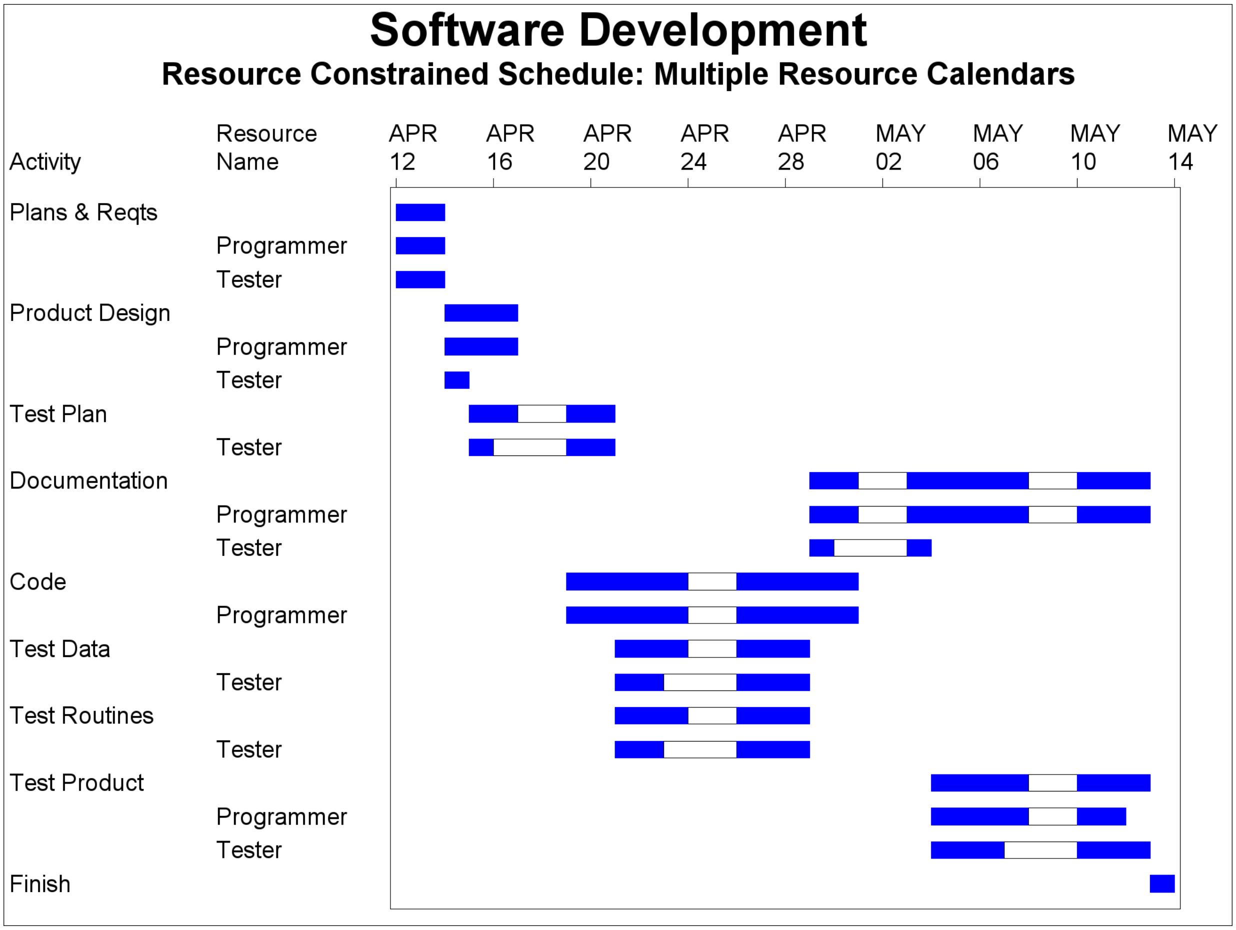
Resource Data Set RESIN2

Obs	per	otype	Programmer	Tester
1	.	calendar	0	1
2	12APR04	reslevel	1	1

Next, invoke PROC CPM, as shown in the following statements, with the Activity, Resource, and Calendar data sets to obtain the revised schedule, plotted in Output 4.24.10. The project is delayed by two days because of the TESTER's shorter work week, which is illustrated by the longer holiday breaks in the TESTER's schedule bars. The new resource constrained schedule is displayed in Output 4.24.11.

```
proc cpm data=software resin=resin2
        caledata=calendar
        out=sftout2 rsched=rsftout2
        resout=rout2
        date='12apr04'd interval=weekday;
  act act;
  succ s1 s2;
  dur dur;
  res Programmer Tester / work=mandays addcal
                          obstype=otype
                          period=per
                          rschedid=Activity;
  id Activity;
run;
```

Output 4.24.10 Resource-Constrained Schedule



Output 4.24.11 Resource-Constrained Schedule: Multiple Calendars

Software Development
Resource Constrained Schedule: Multiple Resource Calendars

Activity	act	_CAL_	RESOURCE	DUR_TYPE	dur	mandays	R_RATE	S_START	S_FINISH
Plans & Reqts	1	0			2	.	.	12APR04	13APR04
Plans & Reqts	1	0	Programmer	FIXED	2	.	1.0	12APR04	13APR04
Plans & Reqts	1	1	Tester	FIXED	2	.	1.0	12APR04	13APR04
Product Design	2	0			3	.	.	14APR04	16APR04
Product Design	2	0	Programmer	RDRIVEN	3	3	1.0	14APR04	16APR04
Product Design	2	1	Tester	RDRIVEN	1	1	1.0	14APR04	14APR04
Test Plan	3	0			3	.	.	15APR04	20APR04
Test Plan	3	1	Tester	FIXED	3	.	1.0	15APR04	20APR04
Documentation	4	0			10	.	.	29APR04	12MAY04
Documentation	4	0	Programmer	RDRIVEN	10	2	0.2	29APR04	12MAY04
Documentation	4	1	Tester	RDRIVEN	2	1	0.5	29APR04	03MAY04
Code	5	0			10	.	.	19APR04	30APR04
Code	5	0	Programmer	FIXED	10	.	0.8	19APR04	30APR04
Test Data	6	0			5	.	.	21APR04	28APR04
Test Data	6	1	Tester	FIXED	5	.	0.5	21APR04	28APR04
Test Routines	7	0			5	.	.	21APR04	28APR04
Test Routines	7	1	Tester	FIXED	5	.	0.5	21APR04	28APR04
Test Product	8	0			6	.	.	04MAY04	12MAY04
Test Product	8	0	Programmer	FIXED	6	.	0.5	04MAY04	11MAY04
Test Product	8	1	Tester	FIXED	6	.	1.0	04MAY04	12MAY04
Finish	9	0			0	.	.	13MAY04	13MAY04

Activity	E_START	E_FINISH	L_START	L_FINISH
Plans & Reqts	12APR04	13APR04	12APR04	13APR04
Plans & Reqts	12APR04	13APR04	12APR04	13APR04
Plans & Reqts	12APR04	13APR04	12APR04	13APR04
Product Design	14APR04	16APR04	14APR04	16APR04
Product Design	14APR04	16APR04	14APR04	16APR04
Product Design	14APR04	14APR04	15APR04	15APR04
Test Plan	14APR04	19APR04	19APR04	21APR04
Test Plan	14APR04	19APR04	19APR04	21APR04
Documentation	19APR04	30APR04	28APR04	11MAY04
Documentation	19APR04	30APR04	28APR04	11MAY04
Documentation	19APR04	20APR04	10MAY04	11MAY04
Code	19APR04	30APR04	19APR04	30APR04
Code	19APR04	30APR04	19APR04	30APR04
Test Data	20APR04	27APR04	22APR04	30APR04
Test Data	20APR04	27APR04	22APR04	29APR04
Test Routines	20APR04	27APR04	22APR04	30APR04
Test Routines	20APR04	27APR04	22APR04	29APR04
Test Product	03MAY04	11MAY04	03MAY04	11MAY04
Test Product	03MAY04	10MAY04	04MAY04	11MAY04
Test Product	03MAY04	11MAY04	03MAY04	11MAY04
Finish	12MAY04	12MAY04	12MAY04	12MAY04

Example 4.25: Resource-Driven Durations and Alternate Resources

Consider the software project defined in Example 4.24 but now the project requires a single resource: a programmer. A network diagram displaying the activities and their precedence relationships is shown in Output 4.24.1, as part of the same example.

Some of the activities in this project have a fixed duration, requiring a fixed length of time from a programmer. Other activities specify the amount of work required in terms of man-days; for these activities, the length of the task will depend on the number of programmers (or rate) that is assigned to the task. The activities in the project, their durations (if fixed) or the total work required (if resource-driven) in days, the precedence constraints, and the resource requirements are displayed in Output 4.25.1.

Suppose that you have only one programmer assigned to the project. You can determine a resource-constrained schedule using PROC CPM by specifying a resource availability data set, resin (also in Output 4.25.1). The Resource data set indicates that the resource Programmer is a driving resource whenever the WORK variable has a valid value.

Output 4.25.1 Project Data

Software Development
Activity Data Set SOFTWARE

Activity	act	s1	s2	dur	mandays	Programmer
Plans & Reqts	1	2	3	2	.	1
Product Design	2	4	5	.	3	1
Test Plan	3	6	7	3	.	.
Documentation	4	9	.	1	2	1
Code	5	8	.	1	10	1
Test Data	6	8	.	5	.	.
Test Routines	7	8	.	5	.	.
Test Product	8	9	.	6	.	1
Finish	9	.	.	0	.	.

Software Development
Resource Availability Data Set

Obs	per	otype	Programmer
1	.	resrcdur	1
2	12APR04	reslevel	1

The following statements invoke PROC CPM with a WORK= specification on the RESOURCE statement, which identifies (in number of man-days, in this case) the amount of work required from the resource Programmer for each activity. If the WORK variable has a missing value, the activity in that observation is assumed to have a fixed duration. The project is scheduled to start on April 12, 2004, and the activities are assumed to follow a five-day work week. The resulting schedule is displayed in Output 4.25.2. For each activity in the project, the value of the variable DUR_TYPE indicates whether the resource drives the activity's duration ('RDRIVEN') or not ('FIXED').

```
proc cpm data=software
         out=sftout1 resout=rout1
         rsched=rsftout1
         resin=resin
         date='12apr04'd interval=weekday;
   act act;
   succ s1 s2;
   dur dur;
   res Programmer / work=mandays
                    obstype=otype
                    period=per
                    rschedid=Activity;
   id Activity;
   run;

title 'Software Development';
title2 'Resource Constrained Schedule: Single Programmer';
proc print data=rsftout1 heading=h;
   id Activity;
   run;
```

Output 4.25.2 Resource Schedule

Software Development
Resource Constrained Schedule: Single Programmer

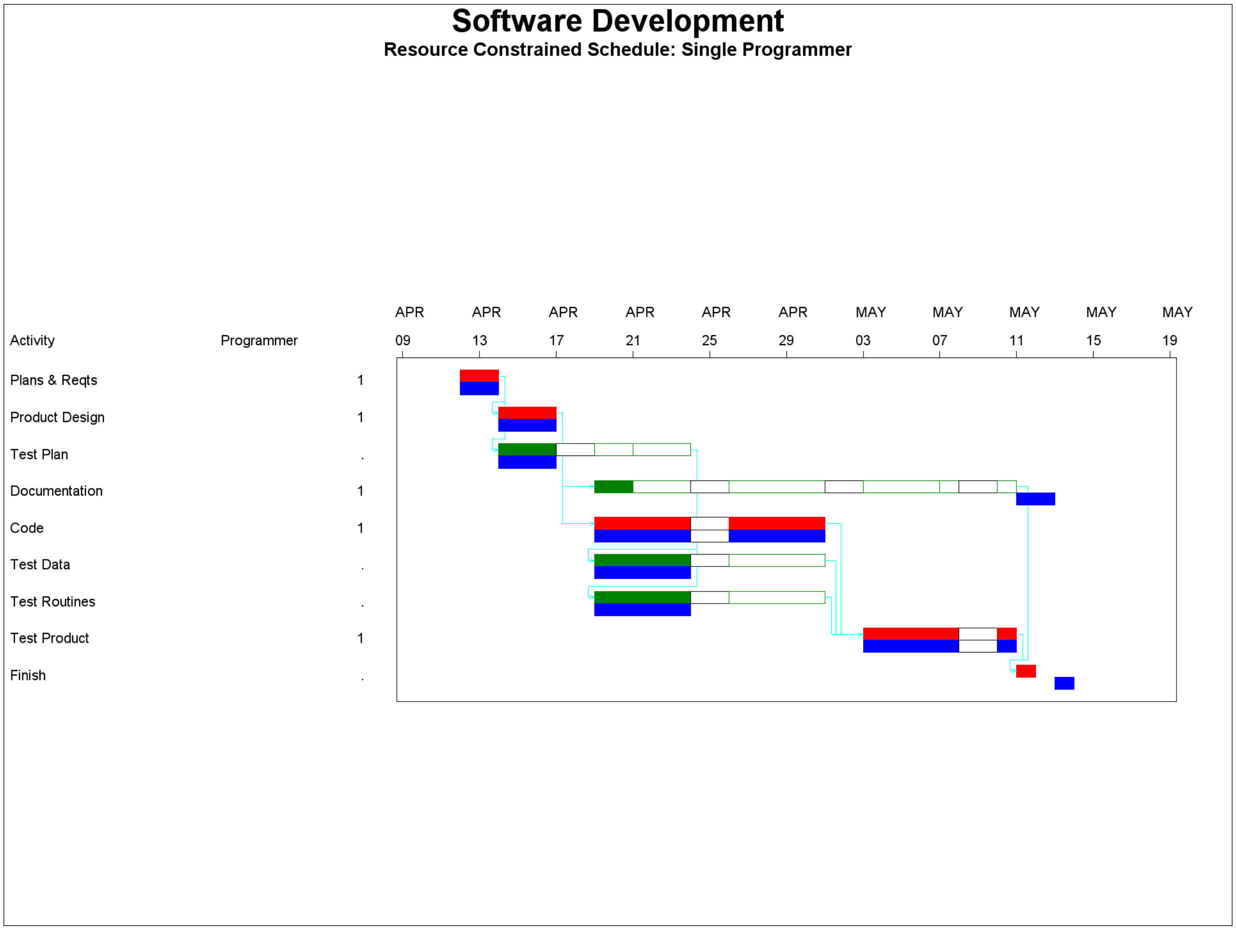
Activity	act	RESOURCE	DUR_TYPE	dur	mandays	R_RATE	S_START	S_FINISH
Plans & Reqts	1			2	.	.	12APR04	13APR04
Plans & Reqts	1	Programmer	FIXED	2	.	1	12APR04	13APR04
Product Design	2			3	.	.	14APR04	16APR04
Product Design	2	Programmer	RDRIVEN	3	3	1	14APR04	16APR04
Test Plan	3			3	.	.	14APR04	16APR04
Documentation	4			1	.	.	11MAY04	12MAY04
Documentation	4	Programmer	RDRIVEN	2	2	1	11MAY04	12MAY04
Code	5			1	.	.	19APR04	30APR04
Code	5	Programmer	RDRIVEN	10	10	1	19APR04	30APR04
Test Data	6			5	.	.	19APR04	23APR04
Test Routines	7			5	.	.	19APR04	23APR04
Test Product	8			6	.	.	03MAY04	10MAY04
Test Product	8	Programmer	FIXED	6	.	1	03MAY04	10MAY04
Finish	9			0	.	.	13MAY04	13MAY04

Activity	E_START	E_FINISH	L_START	L_FINISH
Plans & Reqts	12APR04	13APR04	12APR04	13APR04
Plans & Reqts	12APR04	13APR04	12APR04	13APR04
Product Design	14APR04	16APR04	14APR04	16APR04
Product Design	14APR04	16APR04	14APR04	16APR04
Test Plan	14APR04	16APR04	21APR04	23APR04
Documentation	19APR04	20APR04	07MAY04	10MAY04
Documentation	19APR04	20APR04	07MAY04	10MAY04
Code	19APR04	30APR04	19APR04	30APR04
Code	19APR04	30APR04	19APR04	30APR04
Test Data	19APR04	23APR04	26APR04	30APR04
Test Routines	19APR04	23APR04	26APR04	30APR04
Test Product	03MAY04	10MAY04	03MAY04	10MAY04
Test Product	03MAY04	10MAY04	03MAY04	10MAY04
Finish	11MAY04	11MAY04	11MAY04	11MAY04

The following statements invoke PROC GANTT to display a Gantt chart of the schedule in Output 4.25.3. The activity, ‘Documentation’, is delayed until May 11, 2004, because there is only one programmer available to the project.

```
title h=2.5 'Software Development';
title2 h=1.5 'Resource Constrained Schedule: Single Programmer';
proc gantt graphics data=sftout1;
   id Activity Programmer;
   chart / compress scale=3 increment=4 interval=weekday
           height=2.8 nojobnum nolegend between=5
           act=act succ=(s1 s2)
           cprec=cyan
           caxis=black
           ;
   run;
```

Output 4.25.3 Resource-Constrained Schedule: Single Programmer



Next, suppose that you have two programmers assigned to your project and you can use either one of them for a given task, depending on their availability. To model this scenario, specify Chris and John as alternate resources that can be substituted for the resource Programmer. The Resource data set, resin2, printed in Output 4.25.4, indicates that Chris and John are alternates for Programmer. Specifying an availability of ‘0’ for the resource Programmer ensures that the procedure will assign one of the two programmers, Chris or John, to each task.

The second observation in the data set resin2 indicates two different rates of substitution for the alternate resources. A value less than 1 indicates that the alternate resource is more efficient than the primary resource, while a value greater than 1 indicates that the alternate resource is less efficient. For fixed-duration activities, the use of the alternate resource changes the *rate* of utilization of the resource, while for a resource-driven activity, it changes the *duration* of the resource. The data set resin specifies that John is twice as efficient as the primary resource Programmer while Chris takes one and a half times as long as the generic resource to accomplish a task.

Output 4.25.4 Alternate Programmers

Resource Data Set RESIN2

Obs	per	otype	resid	Programmer	Chris	John
1	.	resrcdur		1	1.0	1.0
2	.	altrate	Programmer	.	1.5	0.5
3	12APR04	reslevel		.	1.0	1.0

The following statements invoke PROC CPM with the new Resource data set and a modified Activity data set that includes the newly added resource variables, Chris and John. You can see the effects of the alternate resource specifications in the Resource Schedule data set, printed in Output 4.25.5. The activity 'Product Design' that takes 3 days of time from a generic programmer actually takes 4.5 days because the programmer used is Chris, who is substituted at a rate of 1.5. On the other hand, the programmer John efficiently completes the task, 'Documentation', in only 1 day, instead of the planned 2 days for a generic programmer. Note also that the start and finish times are specified as SAS datetime values because the substitution of alternate resources results in some of the resource durations being fractional.

```
data software2;
  set software;
  Chris = .;
  John = .;
  run;

proc cpm data=software2 out=sftout2 rsched=rsftout2
         resin=resin2
         date='12apr04'd interval=weekday resout=rout2;
   act act;
   succ s1 s2;
   dur dur;
   res Programmer Chris John / work=mandays
                               obstype=otype
                               period=per
                               resid=resid
                               rschedid=Activity;
   id Activity;
   run;
```

Output 4.25.5 Resource Schedule with Alternate Programmers

Software Development
Resource Constrained Schedule
Alternate Resources at Varying Rates

Activity	act	RESOURCE	DUR_TYPE	dur	mandays	R_RATE	S_START
Plans & Reqts	1			2.0	.	.	12APR04:00:00:00
Plans & Reqts	1	Programmer	FIXED	2.0	.	1.0	.
Plans & Reqts	1	John	FIXED	2.0	.	0.5	12APR04:00:00:00
Product Design	2			3.0	.	.	14APR04:00:00:00
Product Design	2	Programmer	RDRIVEN	3.0	3.0	1.0	.
Product Design	2	Chris	RDRIVEN	4.5	4.5	1.0	14APR04:00:00:00
Test Plan	3			3.0	.	.	14APR04:00:00:00
Documentation	4			1.0	.	.	20APR04:12:00:00
Documentation	4	Programmer	RDRIVEN	2.0	2.0	1.0	.
Documentation	4	John	RDRIVEN	1.0	1.0	1.0	20APR04:12:00:00
Code	5			1.0	.	.	20APR04:12:00:00
Code	5	Programmer	RDRIVEN	10.0	10.0	1.0	.
Code	5	Chris	RDRIVEN	15.0	15.0	1.0	20APR04:12:00:00
Test Data	6			5.0	.	.	19APR04:00:00:00
Test Routines	7			5.0	.	.	19APR04:00:00:00
Test Product	8			6.0	.	.	11MAY04:12:00:00
Test Product	8	Programmer	FIXED	6.0	.	1.0	.
Test Product	8	John	FIXED	6.0	.	0.5	11MAY04:12:00:00
Finish	9			0.0	.	.	19MAY04:12:00:00

Activity	S_FINISH	E_START	E_FINISH
Plans & Reqts	13APR04:23:59:59	12APR04:00:00:00	13APR04:23:59:59
Plans & Reqts	.	12APR04:00:00:00	13APR04:23:59:59
Plans & Reqts	13APR04:23:59:59	.	.
Product Design	20APR04:11:59:59	14APR04:00:00:00	16APR04:23:59:59
Product Design	.	14APR04:00:00:00	16APR04:23:59:59
Product Design	20APR04:11:59:59	.	.
Test Plan	16APR04:23:59:59	14APR04:00:00:00	16APR04:23:59:59
Documentation	21APR04:11:59:59	19APR04:00:00:00	20APR04:23:59:59
Documentation	.	19APR04:00:00:00	20APR04:23:59:59
Documentation	21APR04:11:59:59	.	.
Code	11MAY04:11:59:59	19APR04:00:00:00	30APR04:23:59:59
Code	.	19APR04:00:00:00	30APR04:23:59:59
Code	11MAY04:11:59:59	.	.
Test Data	23APR04:23:59:59	19APR04:00:00:00	23APR04:23:59:59
Test Routines	23APR04:23:59:59	19APR04:00:00:00	23APR04:23:59:59
Test Product	19MAY04:11:59:59	03MAY04:00:00:00	10MAY04:23:59:59
Test Product	.	03MAY04:00:00:00	10MAY04:23:59:59
Test Product	19MAY04:11:59:59	.	.
Finish	19MAY04:12:00:00	11MAY04:00:00:00	11MAY04:00:00:00

Example 4.26: Multiple Alternate Resources

This example illustrates the use of the MULTIPLEALTERNATES option. The Activity data set printed in Output 4.26.1 is a slightly modified version of the data set in Example 4.25. The difference is in the resource requirement for the first activity in the project. The 'Plans and Requirements' task requires 2 programmers. By default, when alternate resources are used, the CPM procedures cannot use multiple alternate resources to substitute for any given resource. In this example, however, you would like the procedure to use both Chris and John for the first task. The Resource data set resmult is also printed in Output 4.26.1, showing that both Chris and John are alternates that can be substituted at the same rate as the primary resource.

Output 4.26.1 Multiple Alternates

Software Development
Use of Multiple Alternate Resources
Activity Data Set

Obs	Activity	dur	mandays	act	s1	s2	Programmer	Chris	John
1	Plans & Reqts	2	.	1	2	3	2	.	.
2	Product Design	.	3	2	4	5	1	.	.
3	Test Plan	3	.	3	6	7	.	.	.
4	Documentation	1	2	4	9	.	1	.	.
5	Code	1	10	5	8	.	1	.	.
6	Test Data	5	.	6	8	.	.	.	.
7	Test Routines	5	.	7	8	.	.	.	.
8	Test Product	6	.	8	9	.	1	.	.
9	Finish	0	.	9	.	.	.	.	.

Software Development
Use of Multiple Alternate Resources
Resource Data Set

Obs	per	otype	resid	Programmer	Chris	John
1	.	resrcdur		1	1	1
2	.	altrate	Programmer	.	1	1
3	12APR04	reslevel		.	1	1

To enable PROC CPM to use multiple alternates, use the MULTIPLEALTERNATES option, as shown in the following invocation:

```
proc cpm data=softmult out=sftmult rsched=rsftmult
         resin=resmult
         date='12apr04'd interval=weekday resout=routmult;
   act act;
   succ s1 s2;
   dur dur;
   res Programmer Chris John / work=mandays
                  obstype=otype
                  period=per resid=resid
                  multiplealternates
                  rschedid=Activity;
   id Activity;
   run;
```

The resulting schedule is printed in Output 4.26.2. Note that both programmers are used for the activity 'Plans and Reqts'.

Output 4.26.2 Multiple Alternates: Resource Schedule Data Set

Software Development
Use of Multiple Alternate Resources
Resource Constrained Schedule

Activity	act	RESOURCE	DUR_TYPE	dur	mandays	R_RATE	S_START	S_FINISH
Plans & Reqts	1			2	.	.	12APR04	13APR04
Plans & Reqts	1	Programmer	FIXED	2	.	2	.	.
Plans & Reqts	1	John	FIXED	2	.	1	12APR04	13APR04
Plans & Reqts	1	Chris	FIXED	2	.	1	12APR04	13APR04
Product Design	2			3	.	.	14APR04	16APR04
Product Design	2	Programmer	RDRIVEN	3	3	1	.	.
Product Design	2	Chris	RDRIVEN	3	3	1	14APR04	16APR04
Test Plan	3			3	.	.	14APR04	16APR04
Documentation	4			1	.	.	19APR04	20APR04
Documentation	4	Programmer	RDRIVEN	2	2	1	.	.
Documentation	4	John	RDRIVEN	2	2	1	19APR04	20APR04
Code	5			1	.	.	19APR04	30APR04
Code	5	Programmer	RDRIVEN	10	10	1	.	.
Code	5	Chris	RDRIVEN	10	10	1	19APR04	30APR04
Test Data	6			5	.	.	19APR04	23APR04
Test Routines	7			5	.	.	19APR04	23APR04
Test Product	8			6	.	.	03MAY04	10MAY04
Test Product	8	Programmer	FIXED	6	.	1	.	.
Test Product	8	Chris	FIXED	6	.	1	03MAY04	10MAY04
Finish	9			0	.	.	11MAY04	11MAY04

Activity	E_START	E_FINISH	L_START	L_FINISH
Plans & Reqts	12APR04	13APR04	12APR04	13APR04
Plans & Reqts	12APR04	13APR04	12APR04	13APR04
Plans & Reqts	.	.	.	.
Plans & Reqts	.	.	.	.
Product Design	14APR04	16APR04	14APR04	16APR04
Product Design	14APR04	16APR04	14APR04	16APR04
Product Design	.	.	.	.
Test Plan	14APR04	16APR04	21APR04	23APR04
Documentation	19APR04	20APR04	07MAY04	10MAY04
Documentation	19APR04	20APR04	07MAY04	10MAY04
Documentation	.	.	.	.
Code	19APR04	30APR04	19APR04	30APR04
Code	19APR04	30APR04	19APR04	30APR04
Code	.	.	.	.
Test Data	19APR04	23APR04	26APR04	30APR04
Test Routines	19APR04	23APR04	26APR04	30APR04
Test Product	03MAY04	10MAY04	03MAY04	10MAY04
Test Product	03MAY04	10MAY04	03MAY04	10MAY04
Test Product	.	.	.	.
Finish	11MAY04	11MAY04	11MAY04	11MAY04

Example 4.27: Auxiliary Resources and Alternate Resources

This example illustrates the use of Auxiliary resources. In the earlier examples, the use of alternate resources enabled the allocation of either John or Chris to the programming tasks. Now, suppose that each of the programmers has a different tester, and whenever a particular programmer is scheduled for a given task, his tester also needs to allocate some part of his or her time, say 50 percent, to the same task. To model such a scenario, specify Tester1 and Tester2 as auxiliary resources for Chris and John, respectively. The Activity and Resource data sets are printed in Output 4.27.1. Unlike the earlier examples, all the activities are of fixed-duration.

```
data software;
     input Activity & $15. dur act s1 s2 Programmer;
datalines;
Plans & Reqts    2  1   2  3  1
Product Design   3  2   4  5  1
Test Plan        3  3   6  7  .
Documentation    3  4   9  .  1
Code             10 5   8  .  1
Test Data        5  6   8  .  .
Test Routines    5  7   8  .  .
Test Product     6  8   9  .  1
Finish           0  9   .  .  .
;

data softaux;
   set software;
   Chris = .;
   John  = .;
   Tester1 = .;
   Tester2 = .;
   run;

data resaux;
   input per date7. otype $ resid $ 18-27 Programmer Chris John
         Tester1 Tester2;
   format per date7.;
   datalines;
.       altrate  Programmer  .  1  1  .  .
.       auxres   Chris       .  .  .  .5 .
.       auxres   John        .  .  .  .  .5
12apr04 reslevel .           .  1  1  1  1
;
```

Output 4.27.1 Auxiliary Resources: Input Data Sets

Software Development
Alternate and Auxiliary Resources
Activity Data Set

Obs	Activity	dur	act	s1	s2	Programmer	Chris	John	Tester1	Tester2
1	Plans & Reqts	2	1	2	3	1	.	.	.	.
2	Product Design	3	2	4	5	1	.	.	.	.
3	Test Plan	3	3	6	7	.	.	.	.	.
4	Documentation	3	4	9	.	1	.	.	.	.
5	Code	10	5	8	.	1	.	.	.	.
6	Test Data	5	6	8	.	.	.	.	.	.
7	Test Routines	5	7	8	.	.	.	.	.	.
8	Test Product	6	8	9	.	1	.	.	.	.
9	Finish	0	9	.	.	.	.	.	.	.

Software Development
Alternate and Auxiliary Resources
Resource Data Set

Obs	per	otype	resid	Programmer	Chris	John	Tester1	Tester2
1	.	altrate	Programmer	.	1	1	.	.
2	.	auxres	Chris	.	.	.	0.5	.
3	.	auxres	John	.	.	.	.	0.5
4	12APR04	reslevel		.	1	1	1.0	1.0

The following statements invoke PROC CPM with the appropriate data sets and resource variables. The resulting schedule is printed in Output 4.27.2. Note the auxiliary resources that have been included in the schedule corresponding to each primary resource: Tester1 whenever Chris is used, and Tester2 whenever John is allocated.

```
proc cpm data=softaux out=sftaux rsched=rsftaux resin=resaux
         date='12apr04'd interval=weekday resout=raux;
   act act;
   succ s1 s2;
   dur dur;
   res Programmer Chris John Tester1 Tester2 /
                            obstype=otype
                            period=per resid=resid
                            multalt rschedid=Activity;
   id Activity;
   run;
```

Output 4.27.2 Auxiliary Resources: Resource Schedule Data Set

Software Development: Alternate and Auxiliary Resources

Resource Schedule Data Set

Activity	act	RESOURCE	DUR_TYPE	dur	_WORK_	R_RATE	S_START	S_FINISH
Plans & Reqts	1			2	.	.	12APR04	13APR04
Plans & Reqts	1	Programmer	FIXED	2	.	1.0	.	.
Plans & Reqts	1	Tester1	FIXED	2	.	0.5	12APR04	13APR04
Plans & Reqts	1	Chris	FIXED	2	.	1.0	12APR04	13APR04
Product Design	2			3	.	.	14APR04	16APR04
Product Design	2	Programmer	FIXED	3	.	1.0	.	.
Product Design	2	Tester1	FIXED	3	.	0.5	14APR04	16APR04
Product Design	2	Chris	FIXED	3	.	1.0	14APR04	16APR04
Test Plan	3			3	.	.	14APR04	16APR04
Documentation	4			3	.	.	19APR04	21APR04
Documentation	4	Programmer	FIXED	3	.	1.0	.	.
Documentation	4	Tester2	FIXED	3	.	0.5	19APR04	21APR04
Documentation	4	John	FIXED	3	.	1.0	19APR04	21APR04
Code	5			10	.	.	19APR04	30APR04
Code	5	Programmer	FIXED	10	.	1.0	.	.
Code	5	Tester1	FIXED	10	.	0.5	19APR04	30APR04
Code	5	Chris	FIXED	10	.	1.0	19APR04	30APR04
Test Data	6			5	.	.	19APR04	23APR04
Test Routines	7			5	.	.	19APR04	23APR04
Test Product	8			6	.	.	03MAY04	10MAY04
Test Product	8	Programmer	FIXED	6	.	1.0	.	.

Activity	E_START	E_FINISH	L_START	L_FINISH
Plans & Reqts	12APR04	13APR04	12APR04	13APR04
Plans & Reqts	12APR04	13APR04	12APR04	13APR04
Plans & Reqts	.	.	.	.
Plans & Reqts	.	.	.	.
Product Design	14APR04	16APR04	14APR04	16APR04
Product Design	14APR04	16APR04	14APR04	16APR04
Product Design	.	.	.	.
Product Design	.	.	.	.
Test Plan	14APR04	16APR04	21APR04	23APR04
Documentation	19APR04	21APR04	06MAY04	10MAY04
Documentation	19APR04	21APR04	06MAY04	10MAY04
Documentation	.	.	.	.
Documentation	.	.	.	.
Code	19APR04	30APR04	19APR04	30APR04
Code	19APR04	30APR04	19APR04	30APR04
Code	.	.	.	.
Code	.	.	.	.
Test Data	19APR04	23APR04	26APR04	30APR04
Test Routines	19APR04	23APR04	26APR04	30APR04
Test Product	03MAY04	10MAY04	03MAY04	10MAY04
Test Product	03MAY04	10MAY04	03MAY04	10MAY04

Output 4.27.2 *continued*

Software Development: Alternate and Auxiliary Resources

Resource Schedule Data Set

Activity	act	RESOURCE	DUR_TYPE	dur	_WORK_	R_RATE	S_START	S_FINISH
Test Product	8	Tester1	FIXED	6	.	0.5	03MAY04	10MAY04
Test Product	8	Chris	FIXED	6	.	1.0	03MAY04	10MAY04
Finish	9			0	.	.	11MAY04	11MAY04

Activity	E_START	E_FINISH	L_START	L_FINISH
Test Product	.	.	.	.
Test Product	.	.	.	.
Finish	11MAY04	11MAY04	11MAY04	11MAY04

Example 4.28: Use of the SETFINISHMILESTONE Option

A simple activity network is used to illustrate the use of the SETFINISHMILESTONE option in a couple of different scenarios.

The following DATA step reads the project network in AON format into a SAS data set named tasks. The data set (printed in Output 4.28.1) contains an Activity variable (act), a Successor variable (succ), a Lag variable (lag), and a Duration variable (dur). There are several milestones linked to other activities through different types of precedence constraints. The data set also contains some alignment constraints as specified by the variables target and trgttype. The treatment of the milestones will vary depending on the presence or absence of the alignment constraints. The data set also contains two variables that indicate the expected early schedule dates for the milestones corresponding to two different invocations of PROC CPM: the variable notrgtmd corresponds to the non-aligned schedule and the variable miledate corresponds to an invocation with the ALIGNDATE statement (the values for these variables are explained later).

```
data tasks;
   format act $7. succ $7. lag $4. target date7.
          trgttype $3. miledate date7. notrgtmd date7. ;
   input act & succ & lag $ dur target & date7.
         trgttype $ miledate & date7. notrgtmd & date7. ;
   datalines;
Task 0   Mile 1   ss_0 1 26Jan04  SGE .        .
Mile 1   Task 2   .    0 .        .   26Jan04  26Jan04
Task 2   .        .    1 .        .   .        .
Task 3   Mile 4   .    1 .        .   .        .
Mile 4   .        .    0 .        .   26Jan04  26Jan04
Task 5   Mile 6   .    1 .        .   .        .
Mile 6   Mile 7   FS_1 0 .        .   26Jan04  26Jan04
Mile 7   .        .    0 .        .   27Jan04  27Jan04
Task 8   Mile 9   SS_3 1 .        .   .        .
Mile 9   Mile 10  .    0 .        .   29Jan04  29Jan04
Mile 10  .        .    0 .        .   29Jan04  29Jan04
Task 11  Mile 12  .    2 .        .   .        .
Mile 12  Mile 13  FS_1 0 28Jan04  SGE 28Jan04  27Jan04
Mile 13  .        .    0 .        .   29Jan04  28Jan04
;
```

Output 4.28.1 Input Data Set

Schedule with option SETFINISHMILESTONE
Input Data Set

Obs	act	succ	lag	target	trgttype	miledate	notrgtmd	dur
1	Task 0	Mile 1	ss_0	26JAN04	SGE	.	.	1
2	Mile 1	Task 2		.		26JAN04	26JAN04	0
3	Task 2			.		.	.	1
4	Task 3	Mile 4		.		.	.	1
5	Mile 4			.		26JAN04	26JAN04	0
6	Task 5	Mile 6		.		.	.	1
7	Mile 6	Mile 7	FS_1	.		26JAN04	26JAN04	0
8	Mile 7			.		27JAN04	27JAN04	0
9	Task 8	Mile 9	SS_3	.		.	.	1
10	Mile 9	Mile 10		.		29JAN04	29JAN04	0
11	Mile 10			.		29JAN04	29JAN04	0
12	Task 11	Mile 12		.		.	.	2
13	Mile 12	Mile 13	FS_1	28JAN04	SGE	28JAN04	27JAN04	0
14	Mile 13			.		29JAN04	28JAN04	0

Output 4.28.2 Default Schedule

Schedule with option SETFINISHMILESTONE
Default Schedule

Obs	act	succ	dur	lag	notrgtmd	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	Task 0	Mile 1	1	ss_0	.	26JAN04	26JAN04	28JAN04	28JAN04	2	0
2	Mile 1	Task 2	0		26JAN04	26JAN04	26JAN04	28JAN04	28JAN04	2	0
3	Task 2		1		.	26JAN04	26JAN04	28JAN04	28JAN04	2	2
4	Task 3	Mile 4	1		.	26JAN04	26JAN04	28JAN04	28JAN04	2	0
5	Mile 4		0		26JAN04	27JAN04	27JAN04	29JAN04	29JAN04	2	2
6	Task 5	Mile 6	1		.	26JAN04	26JAN04	27JAN04	27JAN04	1	0
7	Mile 6	Mile 7	0	FS_1	26JAN04	27JAN04	27JAN04	28JAN04	28JAN04	1	0
8	Mile 7		0		27JAN04	28JAN04	28JAN04	29JAN04	29JAN04	1	1
9	Task 8	Mile 9	1	SS_3	.	26JAN04	26JAN04	26JAN04	26JAN04	0	0
10	Mile 9	Mile 10	0		29JAN04	29JAN04	29JAN04	29JAN04	29JAN04	0	0
11	Mile 10		0		29JAN04	29JAN04	29JAN04	29JAN04	29JAN04	0	0
12	Task 11	Mile 12	2		.	26JAN04	27JAN04	26JAN04	27JAN04	0	0
13	Mile 12	Mile 13	0	FS_1	27JAN04	28JAN04	28JAN04	28JAN04	28JAN04	0	0
14	Mile 13		0		28JAN04	29JAN04	29JAN04	29JAN04	29JAN04	0	0

First, the CPM procedure is invoked with the default treatment of milestones. The resulting schedule is printed in Output 4.28.2. Note the dates for the milestones. Compare these dates with the values of the early finish dates of the immediate predecessors.

The default behavior of the CPM procedure defines the start times for milestones to be at the beginning of the day after the predecessor task (with a standard FS_0 relationship) ends. Thus, for example, the activity, 'Mile 4' has E_START=27JAN04 because its predecessor, 'Task 3', has E_FINISH=26JAN04. The interpretation for these dates are that the early finish corresponds to the end of the day, while the early start of the milestone 'Mile 4' corresponds to the beginning of the day. However, in some situations you may want the milestone to be scheduled at the same time as the end of the predecessor activity. In other words, you may wish the early start time of the milestone 'Mile 4' to be displayed as 26JAN04, with the interpretation that this time actually denotes the end of the day, rather than the beginning. See the section "Finish Milestones" on page 98 for details about the treatment of milestones. In the current example, the variable notrgtmd contains the desired milestone schedule dates corresponding to this special treatment of milestones. To obtain these desired dates, you must use the SETFINISHMILESTONE option.

```
/* Schedule the project */
proc cpm data=tasks out=out0
         collapse interval=day
         date='26jan04'd;
   activity act;
   successor succ /lag=(lag);
   duration dur;
   id lag notrgtmd;
   run;

title2 'Default Schedule';
proc print; run;
```

Next, the CPM procedure is invoked with the option SETFINISHMILESTONE and the resulting schedule is printed in Output 4.28.3. Not all milestones are defined to denote the end of the displayed date; such milestones are referred to as finish milestone. The variables EFINMILE and LFINMILE indicate if the milestone is a finish milestone or not, corresponding to the early or late schedule, respectively. For example, the milestone ‘Mile 12’ has E_FINISH = 27JAN04 and the value of EFINMILE is ‘1’, indicating that the activity finishes at the end of the day on January 27, 2004. The milestone ‘Mile 13’ (with a finish-to-start lag of 1 day) finishes at the end of the day on January 28, 2004. In fact, as the late finish schedule indicates, the value of L_FINISH for ‘Mile 13’ (and the project finish time) is the end of the day on 28JAN04. Both the variables EFINMILE and LFINMILE have the same values for all the activities in this example.

```
proc cpm data=tasks out=out1
         collapse interval=day
         date='26jan04'd
         setfinishmilestone;
   activity act;
   successor succ /lag=(lag);
   duration dur;
   id lag notrgtmd;
   run;


title 'Schedule with option SETFINISHMILESTONE';
title2 'No Target Dates';
proc print heading=v;
  id act;
  var succ lag dur notrgtmd e_start e_finish
      l_start l_finish efinmile lfinmile;
  run;
```

Output 4.28.3 Schedule with SETFINISHMILESTONE Option

Schedule with option SETFINISHMILESTONE
No Target Dates

act	succ	lag	dur	notrgtmd	E_START	E_FINISH	L_START	L_FINISH	EFINMILE	LFINMILE
Task 0	Mile 1	ss_0	1	.	26JAN04	26JAN04	28JAN04	28JAN04	.	.
Mile 1	Task 2		0	26JAN04	26JAN04	26JAN04	28JAN04	28JAN04	.	.
Task 2			1	.	26JAN04	26JAN04	28JAN04	28JAN04	.	.
Task 3	Mile 4		1	.	26JAN04	26JAN04	28JAN04	28JAN04	.	.
Mile 4			0	26JAN04	26JAN04	26JAN04	28JAN04	28JAN04	1	1
Task 5	Mile 6		1	.	26JAN04	26JAN04	27JAN04	27JAN04	.	.
Mile 6	Mile 7	FS_1	0	26JAN04	26JAN04	26JAN04	27JAN04	27JAN04	1	1
Mile 7			0	27JAN04	27JAN04	27JAN04	28JAN04	28JAN04	1	1
Task 8	Mile 9	SS_3	1	.	26JAN04	26JAN04	26JAN04	26JAN04	.	.
Mile 9	Mile 10		0	29JAN04	29JAN04	29JAN04	29JAN04	29JAN04	.	.
Mile 10			0	29JAN04	29JAN04	29JAN04	29JAN04	29JAN04	.	.
Task 11	Mile 12		2	.	26JAN04	27JAN04	26JAN04	27JAN04	.	.
Mile 12	Mile 13	FS_1	0	27JAN04	27JAN04	27JAN04	27JAN04	27JAN04	1	1
Mile 13			0	28JAN04	28JAN04	28JAN04	28JAN04	28JAN04	1	1

The next invocation of CPM illustrates the effect of alignment constraints on the milestones. As explained in the section "Finish Milestones" on page 98, imposing an alignment constraint of type SGE on a milestone may change it from a finish milestone to a start milestone (default behavior) as far as the early schedule of the project is concerned. In the following program, the CPM procedure is invoked with the SETFINISH-MILESTONE option and the ALIGNDATE and ALIGNTYPE statements. The resulting schedule is printed in Output 4.28.4. The early schedule of the milestones should now correspond to the values in the variable miledate. Note also that the activities 'Mile 12' and 'Mile 13' are no longer finish milestones, as indicated by missing values for the variable EFINMILE. The 'SGE' alignment constraint with a target date of 28JAN04 moves the milestone 'Mile 12' to the beginning of January 28, 2004, instead of the end of January 27, 2004.

```
proc cpm data=tasks out=out2
         collapse
         interval=day
         date='26jan04'd
         setfinishmilestone;
   activity act;
   successor succ /lag=(lag);
   duration dur;
   aligndate target;
   aligntype trgttype;
   id target trgttype lag miledate;
   run;
```

```
title 'Schedule with option SETFINISHMILESTONE';
title2 'Target Dates change Early Schedule for some Milestones';
proc print heading=v;
  id act;
  var succ lag target trgttype miledate e_start e_finish
      l_start l_finish efinmile lfinmile;
run;
```

Output 4.28.4 Effect of Alignment Constraints

Schedule with option SETFINISHMILESTONE
Target Dates change Early Schedule for some Milestones

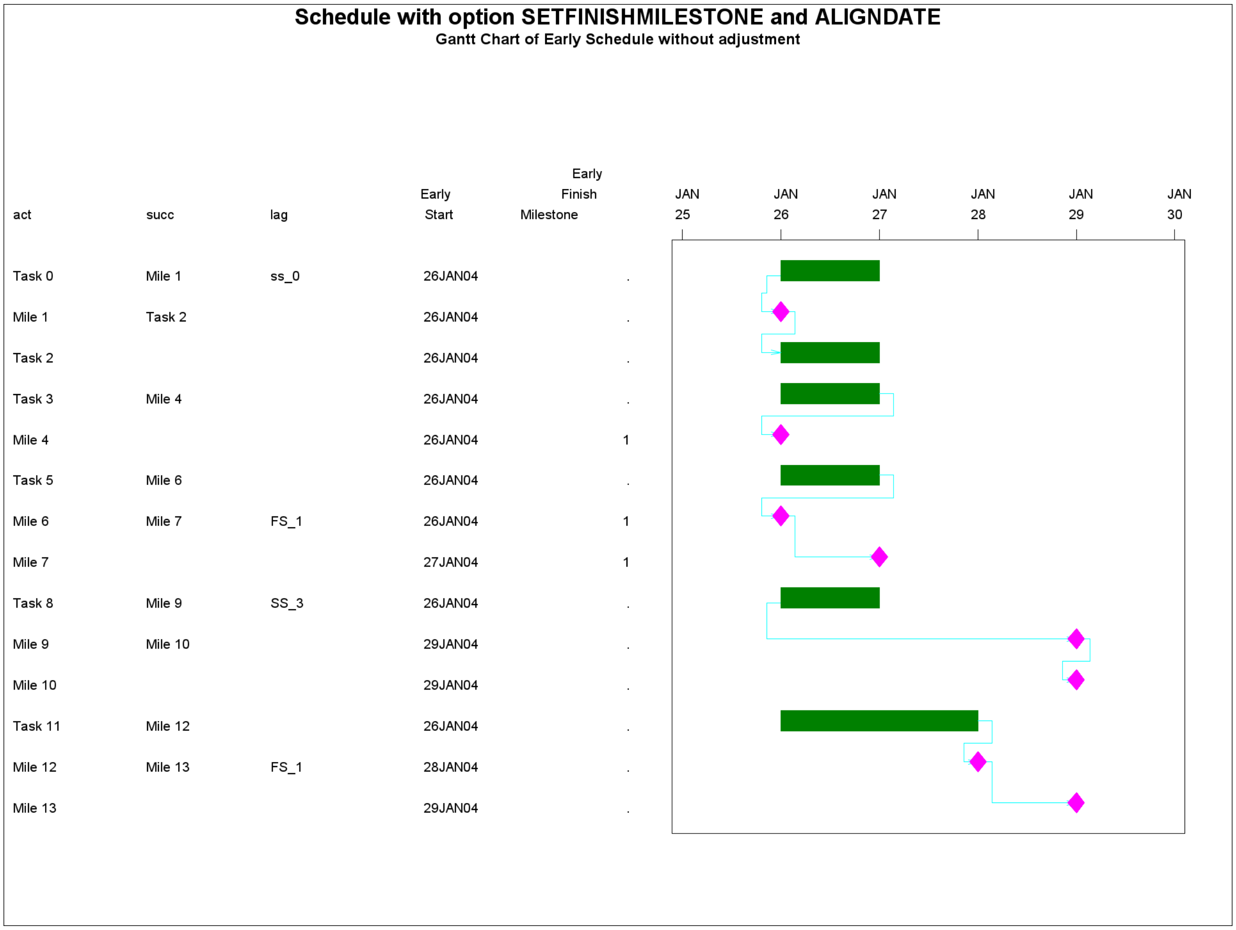
act	succ	lag	target	trgttype	miledate	E_START	E_FINISH	L_START	L_FINISH	EFINMILE	LFINMILE
Task 0	Mile 1	ss_0	26JAN04	SGE	.	26JAN04	26JAN04	28JAN04	28JAN04	.	.
Mile 1	Task 2		.		26JAN04	26JAN04	26JAN04	28JAN04	28JAN04	.	.
Task 2			.		.	26JAN04	26JAN04	28JAN04	28JAN04	.	.
Task 3	Mile 4		.		.	26JAN04	26JAN04	28JAN04	28JAN04	.	.
Mile 4			.		26JAN04	26JAN04	26JAN04	28JAN04	28JAN04	1	1
Task 5	Mile 6		.		.	26JAN04	26JAN04	27JAN04	27JAN04	.	.
Mile 6	Mile 7	FS_1	.		26JAN04	26JAN04	26JAN04	27JAN04	27JAN04	1	1
Mile 7			.		27JAN04	27JAN04	27JAN04	28JAN04	28JAN04	1	1
Task 8	Mile 9	SS_3	.		.	26JAN04	26JAN04	26JAN04	26JAN04	.	.
Mile 9	Mile 10		.		29JAN04	29JAN04	29JAN04	29JAN04	29JAN04	.	.
Mile 10			.		29JAN04	29JAN04	29JAN04	29JAN04	29JAN04	.	.
Task 11	Mile 12		.		.	26JAN04	27JAN04	26JAN04	27JAN04	.	.
Mile 12	Mile 13	FS_1	28JAN04	SGE	28JAN04	28JAN04	28JAN04	27JAN04	27JAN04	.	1
Mile 13			.		29JAN04	29JAN04	29JAN04	28JAN04	28JAN04	.	1

The interpretation of the start and finish times for a milestone depends on whether it is a start milestone or a finish milestone. By default, all milestones are start milestones and are assumed to be scheduled at the beginning of the date specified in the start or finish time variable. As such, PROC GANTT displays these milestones at the start of the corresponding days on the Gantt chart. However, if a milestone is a finish milestone then it may not be displayed correctly on the Gantt chart, depending on the scale of the display.

In this example, PROC GANTT is used to display the schedule produced in Output 4.28.4. Recall that the schedule is saved in the data set out2. First, PROC GANTT is invoked without any modifications to the schedule data set. The resulting Gantt chart is displayed in Output 4.28.5. The finish milestones (with values of EFINMILE = '1') are not plotted correctly. For example, 'Mile 6' is plotted at the *beginning* instead of the *end* of the schedule bar for the predecessor activity, 'Act 5'. To correct this problem, you can adjust the schedule variables for the finish milestones and plot the new values, as illustrated by the second invocation of PROC GANTT. The corrected Gantt chart is displayed in Output 4.28.6.

```
title h=1.5 'Schedule with option SETFINISHMILESTONE and ALIGNDATE';
title2 'Gantt Chart of Early Schedule without adjustment';
proc gantt data=out2(drop=l_:);
   chart / compress act=act succ=succ lag=lag
           scale=7 height=1.7
           cprec=cyan cmile=magenta
           caxis=black
           dur=dur nojobnum nolegend;
   id act succ lag e_start efinmile;
   run;


/* Save adjusted E_START and E_FINISH times for finish
milestones */
data temp;
set out2;
format estart efinish date7.;
estart = e_start;
efinish = e_finish;
if efinmile then do;
   estart=estart+1;
   efinish=efinish+1;
   end;
run;



/* Plot the adjusted start and finish times for the
   early schedule */
title h=1.5 'Schedule with option SETFINISHMILESTONE and ALIGNDATE';
title2 'Gantt Chart of Early Schedule after adjustment';
proc gantt data=temp(drop=l_:);
   chart / compress act=act succ=succ lag=lag
           scale=7 height=1.7
           es=estart ef=efinish
           cprec=cyan cmile=magenta
           caxis=black
           dur=dur nojobnum  nolegend;
   id act succ lag e_start efinmile;
   run;
```

Output 4.28.5 Gantt Chart of Unadjusted Schedule

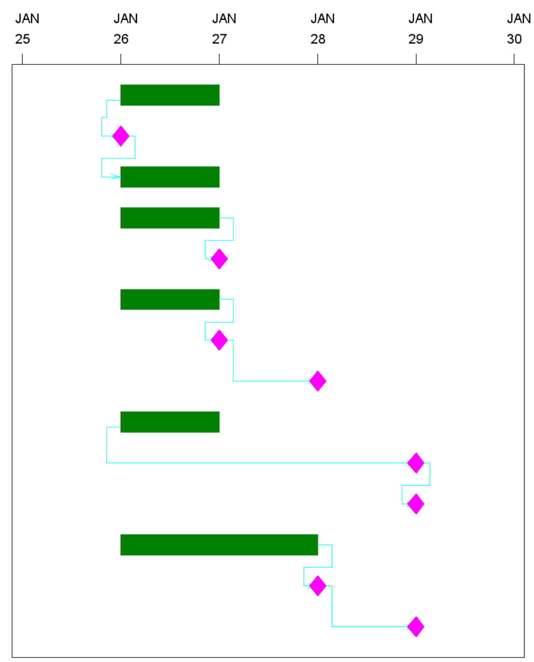


Output 4.28.6 Gantt Chart of Adjusted Schedule

Schedule with option SETFINISHMILESTONE and ALIGNDATE

Gantt Chart of Early Schedule after adjustment

act	succ	lag	Early Start	Early Finish Milestone
Task 0	Mile 1	ss_0	26JAN04	.
Mile 1	Task 2		26JAN04	.
Task 2			26JAN04	.
Task 3	Mile 4		26JAN04	.
Mile 4			26JAN04	1
Task 5	Mile 6		26JAN04	.
Mile 6	Mile 7	FS_1	26JAN04	1
Mile 7			27JAN04	1
Task 8	Mile 9	SS_3	26JAN04	.
Mile 9	Mile 10		29JAN04	.
Mile 10			29JAN04	.
Task 11	Mile 12		26JAN04	.
Mile 12	Mile 13	FS_1	28JAN04	.
Mile 13			29JAN04	.



Example 4.29: Negative Resource Requirements

This example illustrates the use of negative resource requirements and the MILESTONERESOURCE option. Consider the production of boxed greeting cards that need to be shipped on trucks with a given capacity. Suppose there are three trucks with a capacity of 10,000 boxes of cards each. Suppose also that the boxes are produced at the rate of 5,000 boxes a day by the box-creating activity, ‘First Order’ with a duration of 6 days, and requiring the use of a machine, say resource Mach1. The activity data set OneOrder, displayed in Output 4.29.1, represents the activities that are to be scheduled. The “Schedule Truck *i*” task (i = 1, 2, 3) is represented as a milestone to denote the point in time when the required number of boxes are available from the production line. The variable numboxes denotes the number of boxes that are produced by the machine, or delivered by the trucks. The Resource data set OneMachine, displayed in Output 4.29.2, defines the resource numboxes as a consumable resource and the resources Mach1 and trucks as replenishable resources.

Output 4.29.1 Activity Data Set

Negative Resource Requirements
Activity Data Set OneOrder

Obs	Activity	succ	Duration	Mach1	numboxes	trucks
1	First Order		6	1	-5000	.
2	Sched truck1	Delivery 1	0	.	10000	.
3	Sched truck2	Delivery 2	0	.	10000	.
4	Sched truck3	Delivery 3	0	.	10000	.
5	Delivery 1		2	.	.	1
6	Delivery 2		2	.	.	1
7	Delivery 3		2	.	.	1

Output 4.29.2 Resource Data Set

Negative Resource Requirements
Resource Data Set OneMachine

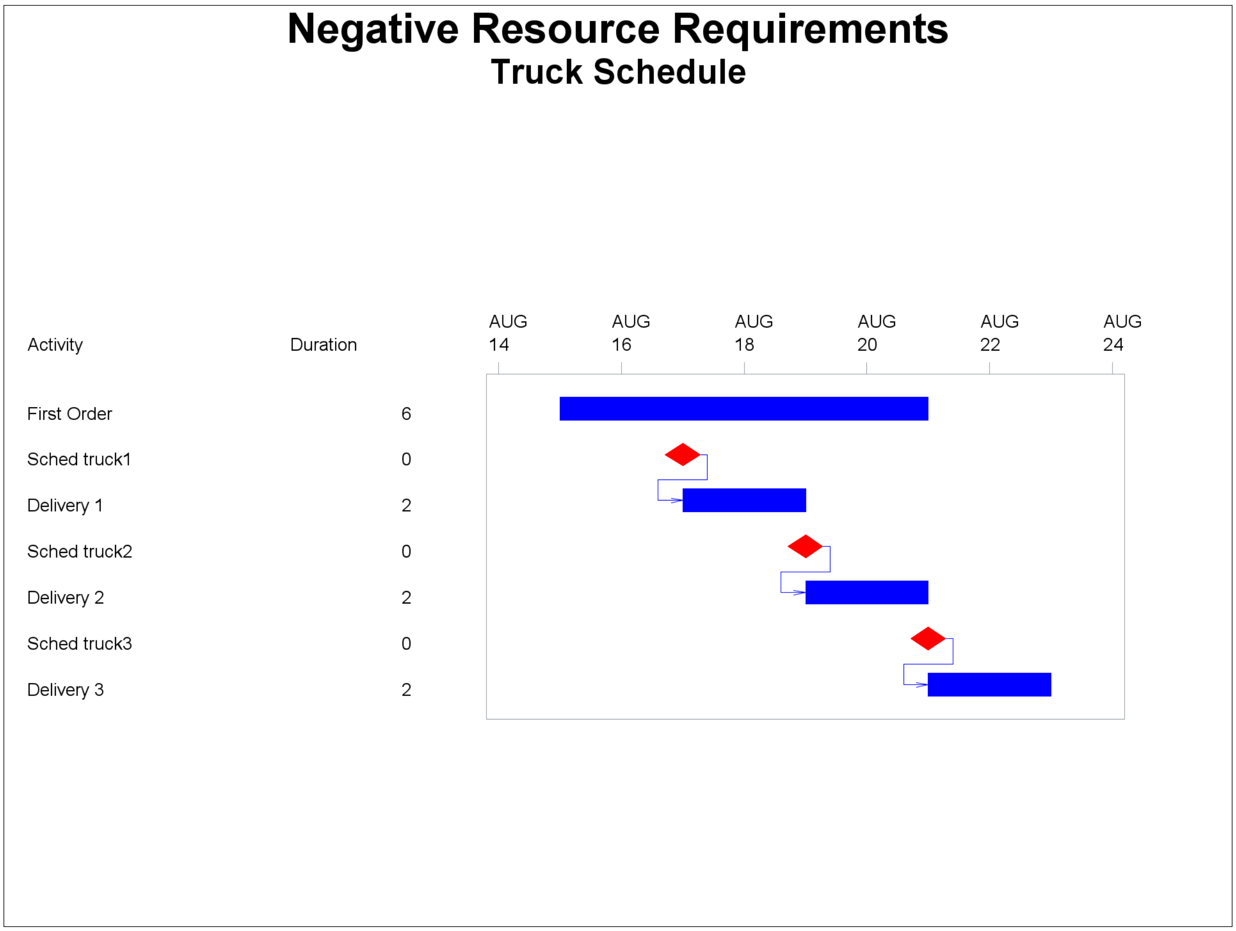
Obs	per	obstype	Mach1	numboxes	trucks
1	.	restype	1	2	1
2	15AUG04	reslevel	1	.	1

The following statements invoke the CPM procedure to schedule the production of the boxed greeting cards. The option MILESTONERESOURCE indicates that milestones can consume resources. In this case, the milestones representing the scheduling of the trucks are scheduled only when 10,000 boxes of greeting cards are available. The resulting schedule is displayed in Output 4.29.3 using PROC GANTT, and the resource usage data set is displayed in Output 4.29.4.

```
proc cpm data=OneOrder resin=OneMachine
     out=OneSched rsched=OneRsch resout=OneRout
     date='15aug04'd;
   act       activity;
   succ      succ;
   duration duration;
   resource Mach1 numboxes trucks / period=per
                                    obstype=obstype
                                    milestoneresource;
   run;

proc sort data=OneSched;
   by s_start;
   run;


title  h=2 'Negative Resource Requirements';
title2 h=1.5   'Truck Schedule';
proc gantt data=OneSched (drop=e_: l_:) ;
   chart / act=activity succ=succ duration=duration
           cmile=red
```

```
            cprec=blue height=1.8
            nolegend nojobnum;
   id activity duration;
   run;


title2 'Resource Usage Data Set';
proc print data=OneRout;
   id _time_;
   run;
```

Output 4.29.3 Gantt Chart of Schedule



The resulting Gantt chart shows the schedule of the trucks, which is staggered according to the production rate of the machine that produces the cards. In other words, the trucks are scheduled at intervals of 2 days. The Resource Usage data set shows the production/consumption rate of the boxes for each day of the project.

Output 4.29.4 Resource Usage Data Set

Resource Usage Data Set

TIME	EMach1	LMach1	RMach1	AMach1	Enumboxes	Lnumboxes	Rnumboxes	Anumboxes
15AUG04	1	1	1	0	25000	-5000	-5000	0
16AUG04	1	1	1	0	-5000	-5000	-5000	5000
17AUG04	1	1	1	0	-5000	-5000	5000	10000
18AUG04	1	1	1	0	-5000	-5000	-5000	5000
19AUG04	1	1	1	0	-5000	25000	5000	10000
20AUG04	1	1	1	0	-5000	-5000	-5000	5000
21AUG04	0	0	0	1	0	0	10000	10000
22AUG04	0	0	0	1	0	0	0	0
23AUG04	0	0	0	1	0	0	0	0

TIME	Etrucks	Ltrucks	Rtrucks	Atrucks
15AUG04	3	0	0	1
16AUG04	3	0	0	1
17AUG04	0	0	1	0
18AUG04	0	0	1	0
19AUG04	0	3	1	0
20AUG04	0	3	1	0
21AUG04	0	0	1	0
22AUG04	0	0	1	0
23AUG04	0	0	0	1

Example 4.30: Auxiliary Resources and Negative Requirements

This example extends the production scenario in the previous example to two separate orders of the greeting cards. Suppose also that the machine used in Example 4.29 is to be replaced by a faster machine that is scheduled to come on-line on August 24, 2004. This scheduling problem is modeled using alternate resources Mach1 and Mach2 for a primary resource Machine. Each of the alternate resources produces the auxiliary resource numboxes; the rate of production depends on which machine is used.

The Activity data set TwoOrders, displayed in Output 4.30.1, now contains additional activities corresponding to the second order of greeting cards. The resource requirement corresponding to the machine needed for the production is now represented in terms of the generic machine resource, Machine. The resource data set, TwoMachines, displayed in Output 4.30.2, specifies Mach1 and Mach2 as alternate resources for Machine and the resource numboxes as an auxiliary resource produced at the rate of 5,000 by Mach1 and 10,000 by Mach2. Observations 5 and 6 indicate that the first machine is available from August 15 and is then replaced by the second machine on August 24, 2004.

Output 4.30.1 Activity Data Set

Auxiliary Resources
Activity Data Set TwoOrder

Obs	Activity	succ	Duration	Machine	Mach1	Mach2	numboxes	trucks	_pattern
1	First Order		6	1	.	.	.	.	1
2	Sched truck1	Delivery 1	0	.	.	.	10000	.	1
3	Sched truck2	Delivery 2	0	.	.	.	10000	.	1
4	Sched truck3	Delivery 3	0	.	.	.	10000	.	1
5	Delivery 1		2	.	.	.	.	1	1
6	Delivery 2		2	.	.	.	.	1	1
7	Delivery 3		2	.	.	.	.	1	1
8	Second Order		6	1	.	.	.	.	2
9	Sched truck4	Delivery 4	0	.	.	.	10000	.	2
10	Sched truck5	Delivery 5	0	.	.	.	10000	.	2
11	Sched truck6	Delivery 6	0	.	.	.	10000	.	2
12	Delivery 4		2	.	.	.	.	1	2
13	Delivery 5		2	.	.	.	.	1	2
14	Delivery 6		2	.	.	.	.	1	2

Output 4.30.2 Resource Data Set

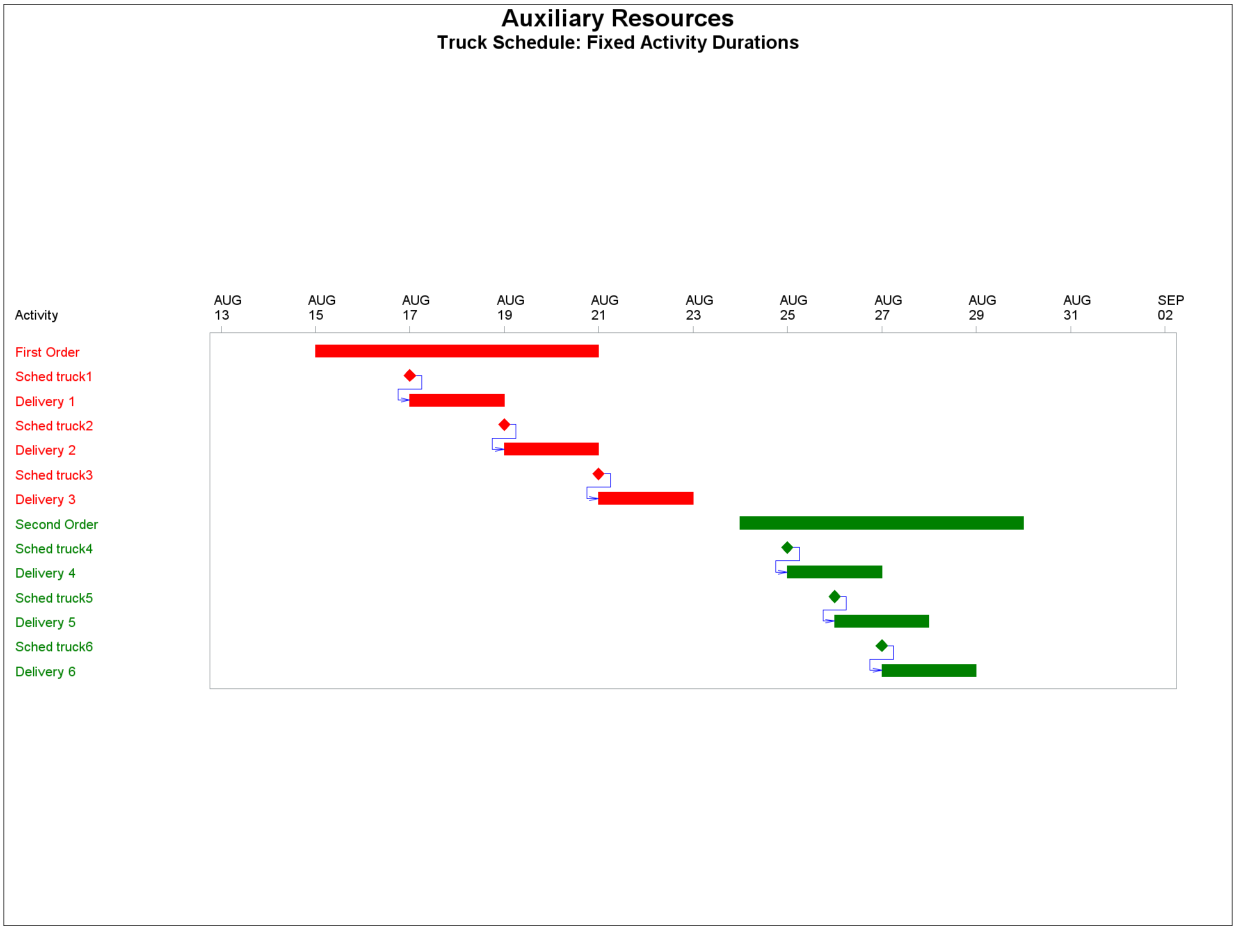
Auxiliary Resources
Resource Data Set TwoMachines

Obs	per	obstype	resid	Machine	Mach1	Mach2	numboxes	trucks
1	.	restype		1	1	1	2	1
2	.	altrate	Machine	.	1	1	.	.
3	.	auxres	Mach1	.	.	.	-5000	.
4	.	auxres	Mach2	.	.	.	-10000	.
5	15AUG04	reslevel		.	1	.	.	3
6	24AUG04	reslevel		.	0	1	.	.

The following statements invoke the CPM procedure to schedule the production of the two orders of boxed greeting cards and display the schedule (in Output 4.30.3) using PROC GANTT. Note that PROC GANTT is invoked with the PATTERN= option indicating that the schedules should be drawn using the pattern statements corresponding to the variable _pattern in the activity data set. In addition, the CTEXTCOLS= option indicates that the color of the text should match the color of the schedule bars.

```
proc cpm data=TwoOrders resin=TwoMachines
     out=TwoSched rsched=TwoRsched resout=TwoRout
     date='15aug04'd;
   act      activity;
   succ     succ;
   duration duration;
   resource Machine Mach1 Mach2 numboxes trucks / period=per
                                         obstype=obstype
                                         resid=resid
                                         milestoneresource;
   id _pattern;
   run;

proc sort data=TwoSched;
   by s_start;
   run;

pattern1 v=s c=red;
pattern2 v=s c=green;

title h=2 'Auxiliary Resources';
title2 h=1.5 'Truck Schedule: Fixed Activity Durations';
proc gantt data=TwoSched(drop=e_: l:);
   chart / act=activity succ=succ duration=duration
           nolegend nojobnum compress pattern=_pattern
           ctextcols=id cprec=blue scale=4 height=1.5;
   id activity ;
   run;

title2 'Resource Usage Data set: Fixed Activity Durations';
proc print data=TwoRout;
   id _time_;
   run;
```

Output 4.30.3 Gantt Chart of Schedule



The Gantt chart shows that the trucks corresponding to the second order of greeting cards depart at a faster rate (every day) than the ones corresponding to the first order (every 2 days). The faster delivery is enabled by the use of the faster machine for the second order. Note also that the activity 'Second Order' continues for a total of 6 days, even though the order is filled within the first 3 days. This is due to the fact that the activity is defined to have a fixed duration. The resource usage data set, displayed in Output 4.30.4 shows that 10,000 boxes are produced each day for 6 days, causing an inventory build up of 30,000 boxes at the end of the production schedule.

Output 4.30.4 Resource Usage Data Set

Auxiliary Resources
Resource Usage Data set: Fixed Activity Durations

TIME	EMachine	LMachine	RMachine	AMachine	EMach1	LMach1	RMach1	AMach1	EMach2	LMach2	RMach2
15AUG04	2	2	0	0	0	0	1	0	0	0	0
16AUG04	2	2	0	0	0	0	1	0	0	0	0
17AUG04	2	2	0	0	0	0	1	0	0	0	0
18AUG04	2	2	0	0	0	0	1	0	0	0	0
19AUG04	2	2	0	0	0	0	1	0	0	0	0
20AUG04	2	2	0	0	0	0	1	0	0	0	0
21AUG04	0	0	0	0	0	0	0	1	0	0	0
22AUG04	0	0	0	0	0	0	0	1	0	0	0
23AUG04	0	0	0	0	0	0	0	1	0	0	0
24AUG04	0	0	0	0	0	0	0	0	0	0	1
25AUG04	0	0	0	0	0	0	0	0	0	0	1
26AUG04	0	0	0	0	0	0	0	0	0	0	1
27AUG04	0	0	0	0	0	0	0	0	0	0	1
28AUG04	0	0	0	0	0	0	0	0	0	0	1
29AUG04	0	0	0	0	0	0	0	0	0	0	1
30AUG04	0	0	0	0	0	0	0	0	0	0	0

TIME	AMach2	Enumboxes	Lnumboxes	Rnumboxes	Anumboxes	Etrucks	Ltrucks	Rtrucks	Atrucks
15AUG04	0	60000	0	-5000	0	6	0	0	3
16AUG04	0	0	0	-5000	5000	6	0	0	3
17AUG04	0	0	0	5000	10000	0	0	1	2
18AUG04	0	0	0	-5000	5000	0	0	1	2
19AUG04	0	0	60000	5000	10000	0	6	1	2
20AUG04	0	0	0	-5000	5000	0	6	1	2
21AUG04	0	0	0	10000	10000	0	0	1	2
22AUG04	0	0	0	0	0	0	0	1	2
23AUG04	0	0	0	0	0	0	0	0	3
24AUG04	0	0	0	-10000	0	0	0	0	3
25AUG04	0	0	0	0	10000	0	0	1	2
26AUG04	0	0	0	0	10000	0	0	2	1
27AUG04	0	0	0	0	10000	0	0	2	1
28AUG04	0	0	0	-10000	10000	0	0	1	2
29AUG04	0	0	0	-10000	20000	0	0	0	3
30AUG04	1	0	0	0	30000	0	0	0	3

Example 4.31: Resource-Driven Durations and Negative Requirements

A more realistic model for the truck scheduling example can be built if the activities 'First Order' and 'Second Order' are defined to be resource driven. In other words, specify the total amount of work (6 days of work) that is needed from the activity at a pre-specified rate (of 5,000 boxes per day), and allow the choice of machine to dictate the duration of the activity. This modified model is illustrated by the activity data set, TwoOrdersRD, and resource data set, TwoMachinesRD, printed in Output 4.31.1 and Output 4.31.1, respectively. The two orders for greeting cards have a work specification of 6 days if the generic machine

Machine (which produces 5,000 boxes a day) is used. The resource data set has a new observation with value 'resrcdur' for the variable obstype. This observation specifies that the resources Machine, Mach1 and Mach2 drive the durations of activities that require them. The third observation in this data set specifies that the second machine is twice as fast as the first one, indicated by the fact that the alternate rate is 0.5. This implies that using the second machine will reduce the activity's duration by 50 percent.

Output 4.31.1 Activity Data Set

Resource-Driven Durations
Activity Data Set TwoOrdersRD

Obs	Activity	succ	Duration	work	Machine	Mach1	Mach2	numboxes	trucks	_pattern
1	First Order		1	6	1	.	.	.	.	1
2	Sched truck1	Delivery 1	0	.	.	.	.	10000	.	1
3	Sched truck2	Delivery 2	0	.	.	.	.	10000	.	1
4	Sched truck3	Delivery 3	0	.	.	.	.	10000	.	1
5	Delivery 1		2	.	.	.	.	.	1	1
6	Delivery 2		2	.	.	.	.	.	1	1
7	Delivery 3		2	.	.	.	.	.	1	1
8	Second Order		1	6	1	.	.	.	.	2
9	Sched truck4	Delivery 4	0	.	.	.	.	10000	.	2
10	Sched truck5	Delivery 5	0	.	.	.	.	10000	.	2
11	Sched truck6	Delivery 6	0	.	.	.	.	10000	.	2
12	Delivery 4		2	.	.	.	.	.	1	2
13	Delivery 5		2	.	.	.	.	.	1	2
14	Delivery 6		2	.	.	.	.	.	1	2

Output 4.31.2 Resource Data Set

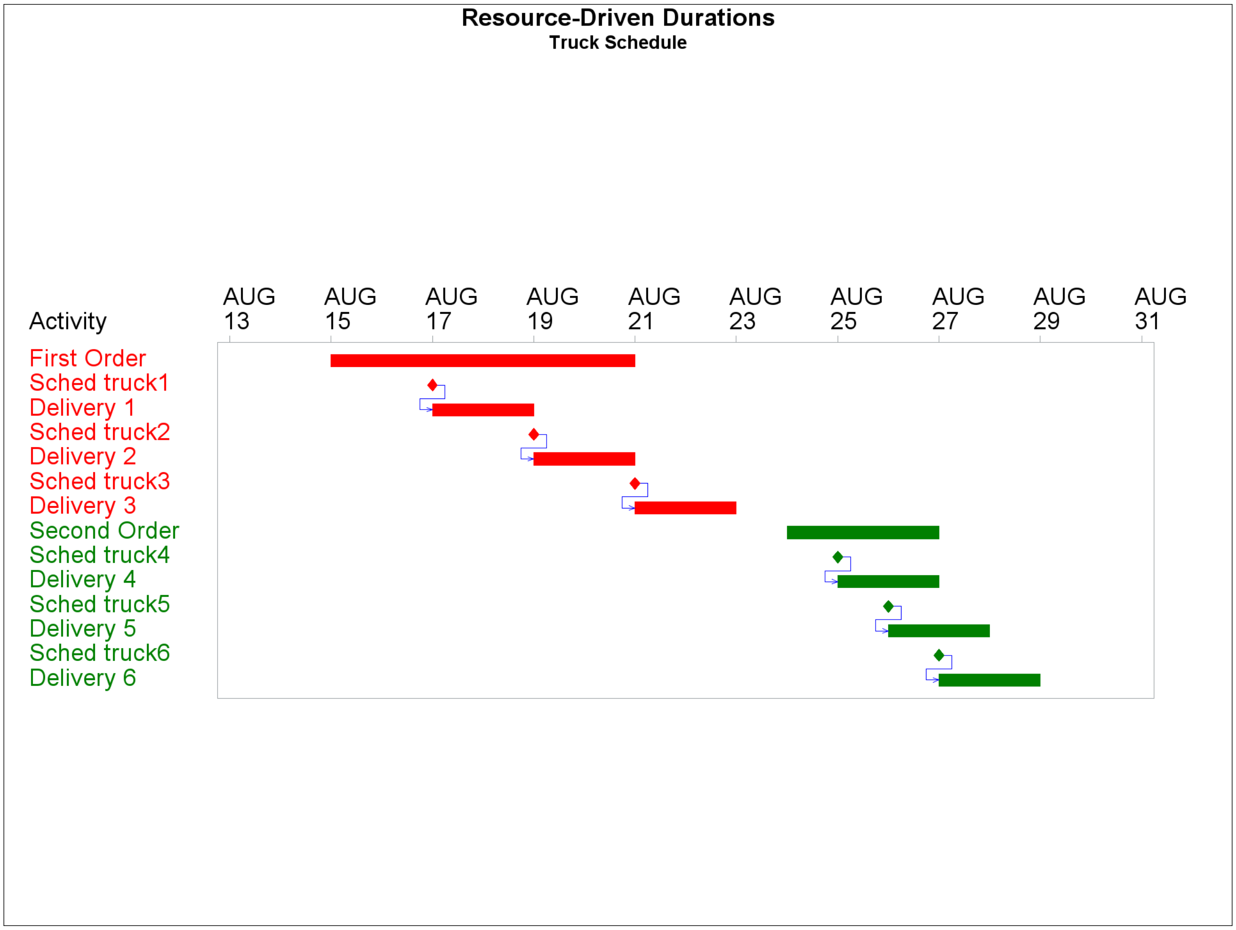
Resource-Driven Durations
Resource Data Set TwoMachinesRD

Obs	per	obstype	resid	Machine	Mach1	Mach2	numboxes	trucks
1	.	resrcdur		1	1	1.0	.	.
2	.	restype		1	1	1.0	2	1
3	.	altrate	Machine	.	1	0.5	.	.
4	.	auxres	Mach1	.	.	.	-5000	.
5	.	auxres	Mach2	.	.	.	-10000	.
6	15AUG04	reslevel		.	1	.	.	3
7	24AUG04	reslevel		.	0	1.0	.	.

The following statements invoke PROC CPM with the additional specification of the WORK= option. Once again, the CPM procedure allocates one of the two machines for the production, depending on the availability. The Gantt chart is displayed in Output 4.31.3 and the resource usage data set is printed in Output 4.31.4. As before, the trucks for the first order depart every second day requiring a total of 6 days, while the second order is completed in 3 days. Also, using a resource-driven duration model allows the second activity to be completed in 3 days instead of 6 days, as in the previous example. The resource usage data set indicates that production is stopped as soon as the two orders are filled, avoiding excess inventory.

```
proc cpm data=TwoOrdersRD resin=TwoMachinesRD
     out=TwoSchedRD rsched=TwoRschedRD resout=TwoRoutRD
     date='15aug04'd;
   act      activity;
   succ     succ;
   duration duration;
   resource Machine Mach1 Mach2 numboxes trucks / period=per
                                         obstype=obstype
                                         resid=resid work=work
                                         milestoneresource;
   id _pattern;
   run;

proc sort data=TwoSchedRD;
   by s_start;
   run;

pattern1 v=s c=red;
pattern2 v=s c=green;

title h=2 'Resource-Driven Durations';
title2 h=1.5 'Truck Schedule';
proc gantt data=TwoSchedRD(drop=e_: l:);
   chart / act=activity succ=succ duration=duration
           nolegend nojobnum compress pattern=_pattern
           ctextcols=id cprec=blue scale=4 height=1.4;
   id activity ;
   run;


title2 'Resource Usage Data Set';
proc print data=TwoRoutRD;
   id _time_;
   run;
```

Output 4.31.3 Gantt Chart of Schedule



Output 4.31.4 Resource Usage Data Set

Resource-Driven Durations
Resource Usage Data Set

TIME	EMachine	LMachine	RMachine	AMachine	EMach1	LMach1	RMach1	AMach1	EMach2	LMach2	RMach2
15AUG04	2	2	0	0	0	0	1	0	0	0	0
16AUG04	2	2	0	0	0	0	1	0	0	0	0
17AUG04	2	2	0	0	0	0	1	0	0	0	0
18AUG04	2	2	0	0	0	0	1	0	0	0	0
19AUG04	2	2	0	0	0	0	1	0	0	0	0
20AUG04	2	2	0	0	0	0	1	0	0	0	0
21AUG04	0	0	0	0	0	0	0	1	0	0	0
22AUG04	0	0	0	0	0	0	0	1	0	0	0
23AUG04	0	0	0	0	0	0	0	1	0	0	0
24AUG04	0	0	0	0	0	0	0	0	0	0	1
25AUG04	0	0	0	0	0	0	0	0	0	0	1
26AUG04	0	0	0	0	0	0	0	0	0	0	1
27AUG04	0	0	0	0	0	0	0	0	0	0	0
28AUG04	0	0	0	0	0	0	0	0	0	0	0
29AUG04	0	0	0	0	0	0	0	0	0	0	0

TIME	AMach2	Enumboxes	Lnumboxes	Rnumboxes	Anumboxes	Etrucks	Ltrucks	Rtrucks	Atrucks
15AUG04	0	60000	0	-5000	0	6	0	0	3
16AUG04	0	0	0	-5000	5000	6	0	0	3
17AUG04	0	0	0	5000	10000	0	0	1	2
18AUG04	0	0	0	-5000	5000	0	0	1	2
19AUG04	0	0	60000	5000	10000	0	6	1	2
20AUG04	0	0	0	-5000	5000	0	6	1	2
21AUG04	0	0	0	10000	10000	0	0	1	2
22AUG04	0	0	0	0	0	0	0	1	2
23AUG04	0	0	0	0	0	0	0	0	3
24AUG04	0	0	0	-10000	0	0	0	0	3
25AUG04	0	0	0	0	10000	0	0	1	2
26AUG04	0	0	0	0	10000	0	0	2	1
27AUG04	1	0	0	10000	10000	0	0	2	1
28AUG04	1	0	0	0	0	0	0	1	2
29AUG04	1	0	0	0	0	0	0	0	3

Statement and Option Cross-Reference Tables

The next two tables reference the statements and options in the CPM procedure that are illustrated by the examples in this section.

Table 4.13 Statements and Options Specified in Examples 2.1–2.17

Statement	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
ACTIVITY	X	X				X	X	X	X	X	X	X	X				
ACTUAL													X				
ALIGNDATE												X					
ALIGNTYPE												X					
BASELINE													X				
CALID										X							
DURATION	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
HEADNODE		X	X	X	X									X	X	X	X
HOLIDAY								X	X	X			X	X	X	X	X
ID	X	X	X	X	X	X								X	X	X	X
RESOURCE														X	X	X	X
SUCCESSOR	X					X	X	X	X	X	X	X	X				
TAILNODE		X	X	X	X									X	X	X	X
Option	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
A_FINISH													X				
ALAGCAL=											X						
A_START=													X				
AUTOUPDT													X				
AVPROFILE																	X
CALEDATA=									X	X	X						
COLLAPSE											X						
COMPARE=													X				
CUMUSAGE																X	X
DATA=	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DATE=	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DAYLENGTH=							X		X	X							
DAYSTART=							X										
DELAY=																X	X
DELAYANALYSIS															X	X	X
FBDATE=			X														
HOLIDATA=								X	X	X			X	X	X	X	X
HOLIDUR=								X		X							
HOLIFIN=								X	X	X			X				
INFEASDIAGNOSTIC																	X
INTERVAL=			X	X	X	X	X	X	X		X	X		X	X	X	X
LAG=											X						
MAXDATE=														X			
NOAUTOUPDT													X				
OBSTYPE=															X	X	X
OUT=		X		X	X	X		X	X	X	X		X		X	X	X
PCTCOMP=													X				
PERIOD=															X	X	X
RCPROFILE																	X
REMDUR=													X				

Table 4.13 (continued)

RESOURCEIN=															X	X	X
RESOURCEOUT=														X	X	X	X
ROUTNOBREAK																X	
SCHEDRULE=															X		
SET=													X				
SHOWFLOAT													X				
TIMENOW=													X				
WORKDATA=									X	X							
XFERVARS												X					

Table 4.14 Statements and Options Specified in Examples 2.18–2.31

Statement	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ACTIVITY		X	X		X	X	X	X	X	X	X	X	X	X
ALIGNDATE						X					X			
ALIGNTYPE						X					X			
BASELINE	X													
DURATION	X	X	X	X	X	X	X	X	X	X	X	X	X	X
HEADNODE	X			X										
HOLIDAY	X	X	X			X								
ID	X			X		X	X	X	X	X	X		X	X
PROJECT						X								
RESOURCE	X	X	X		X		X	X	X	X		X	X	X
SUCCESSOR		X	X		X	X	X	X	X	X	X	X	X	X
TAILNODE	X			X										
Option	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
ACTDELAY=	X													
ADDCAL							X							
ADDWBS						X								
ALTBEFORESUP			X											
AVPROFILE	X	X	X											
CALEDATA=							X							
COLLAPSE		X	X								X			
COMPARE=	X													
CUMUSAGE	X													
DATA=	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DATE=	X	X	X	X		X	X	X	X	X	X	X	X	X
DELAY=			X											
DELAYANALYSIS	X		X											
FBDATE=						X								
F_FLOAT		X												
HOLIDATA=	X	X	X			X								
INFEASDIAGNOSTIC	X													
INTERVAL=	X	X	X			X	X	X	X	X	X			
LAG=											X			
MILESTONERESOURCE												X	X	X
MINSEGMTDUR=		X												
MULTIPLEALTERNATES									X	X				
OBSTYPE=	X	X	X				X	X	X	X		X	X	X
ORDERALL						X								

Table 4.14 (continued)

OUT=	X	X	X	X	X	X	X	X	X	X	X	X	X	X
PERIOD=	X	X	X		X		X	X	X	X		X	X	X
RCPROFILE	X	X	X											
RESID=			X					X	X	X			X	X
RESOURCEIN=	X	X	X		X		X	X	X	X		X	X	X
RESOURCEOUT=	X	X	X				X	X	X	X		X	X	X
RESOURCESCHED=							X	X	X	X		X	X	X
ROUTNOBREAK														
RSCHEDID=							X	X	X	X				
SEPCRIT						X								
SET=	X													
SETFINISHMILESTONE											X			
STOPDATE=					X									
T_FLOAT	X	X												
USEPROJDUR						X								
WORK=							X	X	X					X

References

Davis, E. W. (1973). "Project Scheduling under Resource Constraints: Historical Review and Categorization of Procedures." *AIIE Transactions* 5:297–313.

Elmaghraby, S. E. (1977). *Activity Networks: Project Planning and Control by Network Models*. New York: John Wiley & Sons.

Horowitz, E., and Sahni, S. (1976). *Fundamentals of Data Structures*. Potomac, MD: Computer Science Press.

Kulkarni, R. (1991). "Scheduling with the CPM Procedure." In *Proceedings of the Sixteenth Annual SAS Users Group International Conference*, 700–714. Cary, NC: SAS Institute Inc. `http://www.sascommunity.org/sugi/SUGI91/Sugi-91-118%20Kulkarni.pdf`.

Malcolm, D. G., Roseboom, J. H., Clark, C. E., and Fazar, W. (1959). "Applications of a Technique for R and D Program Evaluation (PERT)." *Operations Research* 7:646–669.

Minieka, E. (1978). *Optimization Algorithms for Networks and Graphs*. New York: Marcel Dekker.

Moder, J. J., Phillips, C. R., and Davis, E. W. (1983). *Project Management with CPM, PERT, and Precedence Diagramming*. New York: Van Nostrand Reinhold.

SAS Institute Inc. (1993). *SAS/OR Software: Project Management Examples*. Cary, NC: SAS Institute Inc.

Van Slyke, R. M. (1963). "Monte Carlo Methods and the PERT Problem." *Operations Research* 11:839–860.

Wiest, J. D. (1967). "A Heuristic Model for Scheduling Large Projects with Limited Resources." *Management Science* 13:359–377.

Chapter 5
The PM Procedure

Contents

Overview: PM Procedure

The PM procedure is an interactive procedure that can be used for planning, controlling, and monitoring a project. The syntax and the scheduling features of PROC PM are virtually the same as those of the CPM procedure. However, because the PM procedure is interactive, there are a few extra options that are available and a few other options that have a default behavior that is different from the CPM procedure. These differences are noted in the section “Syntax: PM Procedure” on page 306 and the section “Summary of Differences” on page 329. One major difference is that only the Activity-On-Node representation of the project is supported in PROC PM. In other words, TAILNODE and HEADNODE statements from PROC CPM are not supported.

For a complete description of the syntax and the scheduling algorithm for the CPM procedure, see Chapter 4, "The CPM Procedure."

When PROC PM is invoked with the activity network representation, an interactive window is opened that displays a Table View of the project on the left and a Gantt View of the project on the right. You can add activities and edit the project data by using the Table View. You can also use the Gantt View to move activities, change the durations of the activities, and add precedence constraints between the activities. These features are described in the section "Details: PM Procedure" on page 309.

The PM procedure is designed to facilitate its inclusion in a Project Management application. Any changes that are made to the activity network or to the activity durations, resource requirements, alignment specifications, and other activity information need to be saved in the resulting Schedule output data set. Further, you should be able to use this output data set as input to a future invocation of PROC PM and continue to manage the project. Thus, there are some differences in the design of the Schedule data set (defined in Chapter 4, "The CPM Procedure") to enable the integration of PROC PM into a Project Management application. The differences between the Schedule data sets in the two procedures are described in the section "Schedule Data Set" on page 329.

Getting Started: PM Procedure

Consider the simple software development project described in the "Getting Started" section of Chapter 4, "The CPM Procedure." Recall that the Activity data set, SOFTWARE, contains the activity descriptions, durations, and precedence constraints. The following statements (identical to the PROC CPM invocation) initialize the project data and invoke the PM procedure.

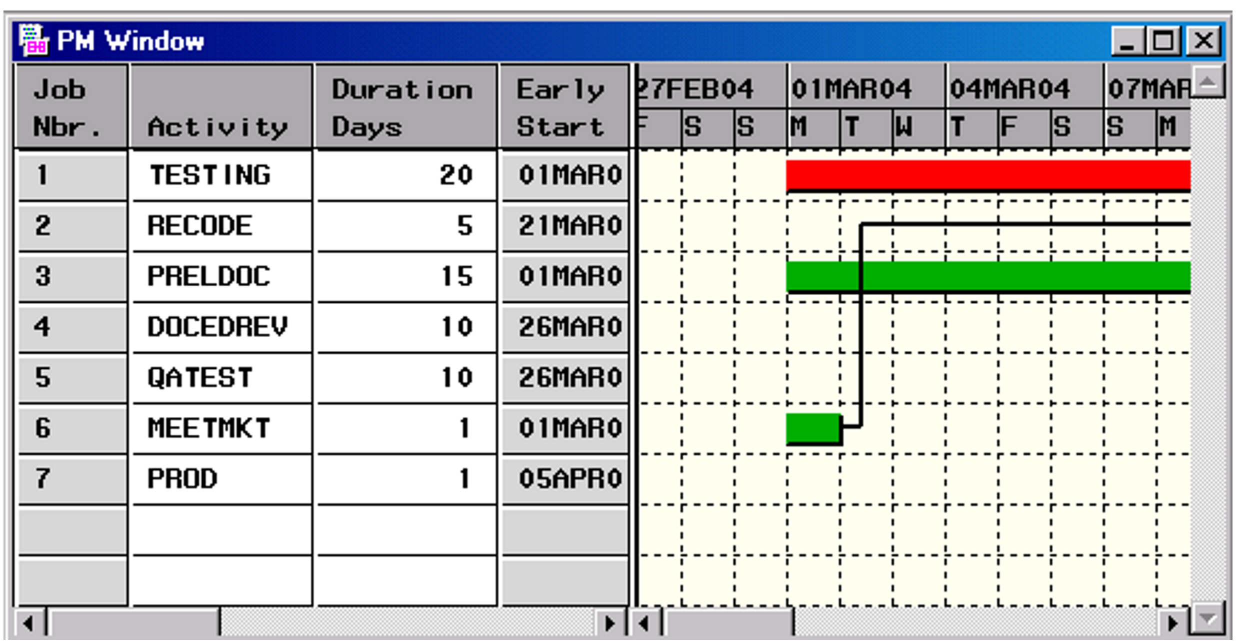
```
data software;
   input Descrpt  $char20.
         Duration 23-24
         Activity $ 27-34
         Succesr1 $ 37-44
         Succesr2 $ 47-54;
   datalines;
Initial Testing       20  TESTING   RECODE
Prel. Documentation   15  PRELDOC   DOCEDREV  QATEST
Meet Marketing        1   MEETMKT   RECODE
Recoding              5   RECODE    DOCEDREV  QATEST
QA Test Approve       10  QATEST    PROD
Doc. Edit and Revise  10  DOCEDREV  PROD
Production            1   PROD
;
```

```
proc pm data=software
         out=intro1
         interval=day
         date='01mar04'd;
   id descrpt;
   activity activity;
   duration duration;
   successor succesr1 succesr2;
run;
```

When you invoke the PM procedure, the PM window appears (see Figure 5.1), consisting of the Table View and the Gantt View of the project. The activities are listed in the order in which they are defined in the Activity data set. The two views are separated by a dividing line that can be dragged to the left or right, controlling the size of the two views. Further, the two views scroll together in the vertical direction but can scroll independently in the horizontal direction.

The Table View contains several editable columns (in white) that can be used to edit the project data as well as add new activities to the project. Some of the columns (in gray), such as the Schedule times, are not editable. The Gantt View contains a Gantt chart of the project and displays the precedence relationships between the activities. You can use the Gantt View to add or delete precedence constraints between activities and to change the durations or alignment constraints of the activities by dragging the schedule bars. Details of the interface are described in the section "Details: PM Procedure" on page 309.

Figure 5.1 Software Development Project



Syntax: PM Procedure

The syntax for PROC PM is virtually identical to that for PROC CPM. The main difference is that you replace the PROC CPM statement with the PROC PM statement.

The TAILNODE and HEADNODE statements from PROC CPM are not supported in PROC PM.

The form of the PROC PM statement is

PROC PM *options* **;**

PROC PM Statement

PROC PM *options* **;**

All the options that are available in the PROC CPM statement can also be specified in the PROC PM statement. See Chapter 4, "The CPM Procedure," for details. However, there are a few additional options available with PROC PM, and some of the other PROC CPM options are not needed as they are the default behavior in PROC PM. See "Summary of Differences" on page 329 for more details about these differences.

Options Specific to PROC PM

The following options can be specified on the PROC PM statement.

NODISPLAY

invokes the procedure in a noninteractive mode. The schedule for the project is still computed and the requested output data sets are created and saved. However, the PM window is not displayed. This option is useful for scheduling large projects that do not need to be updated interactively. Note that invoking PROC PM with the NODISPLAY option is similar to invoking PROC CPM; however, because the format of the Schedule output data set is different for the two procedures, you might see some differences in the order and content of the observations. See "Schedule Data Set" on page 329 for details.

PROJECT=*SAS-data-set*

identifies a SAS data set that can be used to save and restore preferences that control the project view. For example, preferences such as the font, column order, column widths, filters, and so forth, can be saved from one invocation to another. See "PROJECT Data Set" on page 325 for more details about this data set and the preferences that can be saved in it.

PROJECTNAME=*'string'*

PROJNAME=*'string'*

NAME=*'string'*

specifies a descriptive string identifying the name of the project. This string is used to label the PM window.

SUMMARYNAME='*string*'

SUMMARY='*string*'

PROJECTSUMMARY='*string*'

specifies a descriptive string identifying the summary task. By default, when there is more than one root parent activity in a project, PROC PM creates a summary task named "Summary" (or "Project Summary" if the input format for the activity variable is 15 or greater). So, if there is already a child activity named "Summary" (or "Project Summary") in the input data, the resulting schedule forms a cycle. The SUMMARYNAME= option enables you to override the default by specifying a different name for the summary task, thereby avoiding the previously described problem.

Default Options for PROC PM Statement

The following options of PROC CPM are turned on by default in PROC PM.

ADDACT

ADDALLACT

EXPAND

indicates that an observation is to be added to the Schedule output data set (and the Resource Schedule output data set) for each activity that appears as a value of the variables specified in the SUCCESSOR or PROJECT statements without appearing as a value of the variable specified in the ACTIVITY statement. In other words, the Schedule output data set produced by PROC PM contains one observation for every activity that appears as a value of the ACTIVITY, SUCCESSOR, or PROJECT variables (as long as it has not been deleted in the current invocation of the procedure). It also contains an observation for every activity that is added to the project using the graphical user interface.

XFERVARS

indicates that all relevant variables are to be copied from the Activity data set to the Schedule data set. The procedure carries over to the output data set all the relevant variables from the input data set. Thus, the Schedule output data set contains all the project information that is necessary to schedule it.

Default Options for ACTUAL Statement

AUTOUPDT

requests that the procedure assume automatic completion (or start) of activities that are predecessors to activities already completed (or in progress).

ESTIMATEPCTC

ESTPCTC

ESTPCTCOMP

ESTPROG

indicates that a variable named PCT_COMP is to be added to the Schedule output data set (and the Resource Schedule output data set) that contains the percent completion time for each activity (for each resource used by each activity) in the project.

SHOWFLOAT

indicates that activities that are completed or in progress have nonzero float.

Default Options for PROJECT Statement

ADDWBS

WBSCODE

WBS

indicates that the PM procedure is to compute a WBS code for the activities in the project using the project hierarchy structure specified. This code is computed for each activity and stored in the variable WBS_CODE in the Schedule output data set.

DESCENDING

DESC

indicates that, in addition to the ascending sort variables (ES_ASC, LS_ASC, and SS_ASC) that are requested by the ESORDER, LSORDER, and SSORDER options, the corresponding descending sort variables (ES_DESC, LS_DESC, and SS_DESC, respectively) are also to be added to the Schedule output data set.

ESORDER

ESO

indicates that a variable named ES_ASC is to be added to the Schedule output data set; this variable can be used to order the activities in such a way that the activities within each subproject are in increasing order of the early start time. Note that this order is not necessarily the same as the one that would be obtained by sorting all the activities in the Schedule data set by E_START.

LSORDER

LSO

indicates that a variable named LS_ASC is to be added to the Schedule output data set; this variable can be used to order the activities in such a way that the activities within each subproject are in increasing order of the late start time.

ORDERALL

ALL

is equivalent to specifying the ESORDER and LSORDER options (and the SSORDER option when resource constrained scheduling is performed).

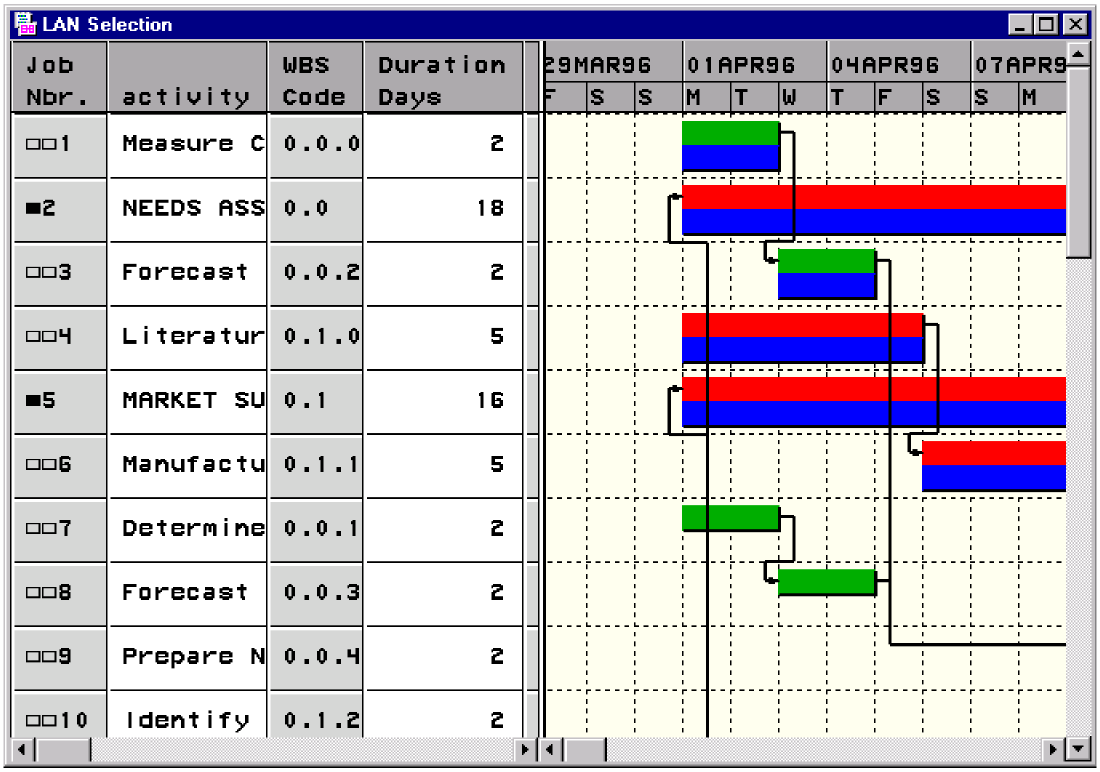
SSORDER

SSO

indicates that a variable named SS_ASC is to be added to the Schedule output data set; this variable can be used to order the activities in such a way that the activities within each subproject are in increasing order of the resource-constrained start time.

Details: PM Procedure

Figure 5.2 LAN Selection Project



The PM window provides the standard editing and viewing functions of a typical project management tool. It can be displayed by invoking the PM procedure. For an existing project, the PM window is populated with the activities in the project. For a new project, the PM window is empty. Figure 5.2 displays the PM window for a sample project.

After you have finished editing the project, you can close the PM window to save the new project data in the Schedule output data set that was specified in the invocation of the PM procedure.

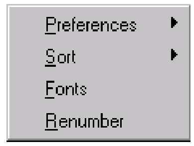
User Interface Features

This section describes some of the typical features of the PM window's graphical user interface. The PM window provides both a Gantt View and a Table View of the project. The size of each view can be changed by pointing to the dividing line between the two views until the pointer changes to a double arrow and then dragging it to the right or left.

Only part of the project may be visible in the PM window; horizontal and vertical scroll bars enable you to scroll the project data in both directions. Note that the Gantt and the Table Views are attached to each other so that they scroll together vertically. Each view can be scrolled horizontally, independently of the other.

The menu associated with the PM window provides access to several project management functions under the **Edit**, **View**, and **Project** menus. For example, the **Project** menu is shown in Figure 5.3. The commands available through the menus are described in detail in the appropriate sections.

Figure 5.3 Project Menu

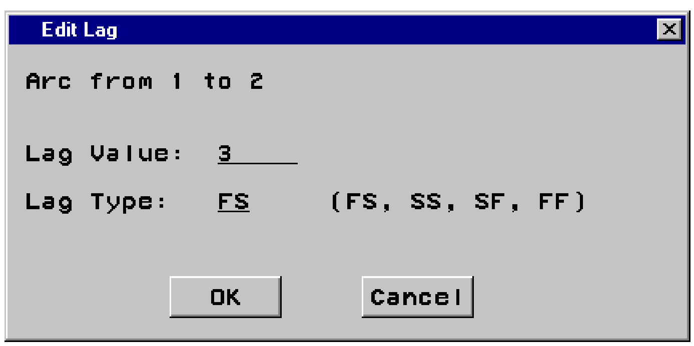


In addition to the drop-down menus, context-sensitive pop-up menus are available in the Table and Gantt Views, the time axis, along the arcs, and from select columns in the Table View. You open a pop-up menu by right-clicking on a particular object. For example, right-clicking on an arc in the Gantt View displays the arc pop-up menu shown in Figure 5.4.

Figure 5.4 Arc Pop-up Menu

In some situations, the pop-up menu selection can lead to a dialog box that requires you to type a value in one or more of the fields in the box. For example, selecting **Edit Lag** from the arc pop-up menu leads to the dialog box displayed in Figure 5.5. (See “Create Nonstandard Precedence Relationships” on page 323 for a discussion of nonstandard precedence constraints.)

Figure 5.5 Edit Lag Dialog Box



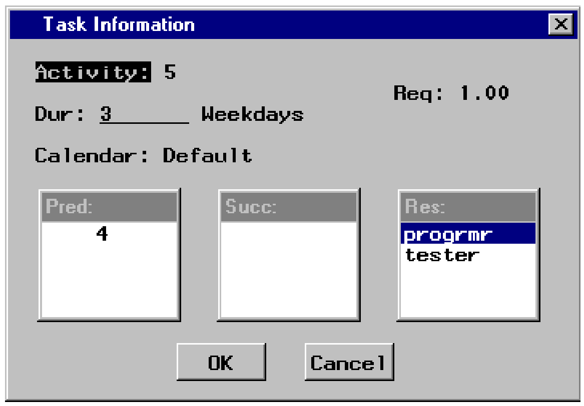
The Table View displays project data in a tabular format. Some of the columns are editable (white background) while other columns, which are computed by the procedure, are not editable (gray background). The Gantt View always displays the early start schedule of the project. In addition, it also displays the resource-constrained schedule (if resources are present), the actual schedule (if the project has started and is in progress), and the baseline schedule (if a baseline schedule is saved for the project). The display of all the schedule bars (except the Early Schedule bar) can be toggled on or off using the pop-up menu from the Gantt View.

Note that each row of the combined Table View and Gantt View represents one activity (also referred to as *task* in this chapter). Any change in data or movement of a row in one view is also reflected in the other.

In addition to the drop-down and pop-up menu actions, several drag-and-drop actions are available within the PM window. You can move the columns and rows of the Table View by selecting a row or column and dragging it to the desired position. You can also change the width of the columns by dragging the column dividers in the Table header region.

You can manipulate the durations of the tasks using the **Task Information** dialog box (see Figure 5.6) by right-clicking on the bar shown in the Gantt View or by changing the length of the Early Schedule bar in the Gantt View. You can also move the task in time by dragging the Early Schedule bar to a new position. This affects the Target Date for the associated task.

Figure 5.6 Task Information Dialog Box



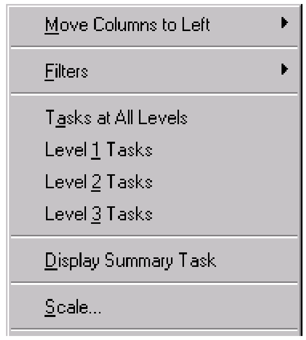
Any of the preceding actions may result in a change to the project schedule that is immediately reflected in the Table and Gantt Views. All editing abilities and the corresponding changes to the schedule are described in detail in the following sections.

Project Hierarchy

The PM procedure displays a hierarchical project structure if it is invoked with the PROJECT statement. If the procedure is invoked without a PROJECT statement, the supertask and subtask relationship is not supported, and all the activities are considered to be at the same level, belonging to a single project.

If the PROJECT statement is used, then a task's level in the project hierarchy is indicated in the Table View by small square boxes to the left of the activity number in the Job Nbr. column. Empty boxes indicate that the activity does not have any subtasks (it is a *leaf* activity), while filled boxes indicate that the activity is a supertask. Further, a Project Summary task is included to represent the root task (or Summary Task) of the project. This task is positioned at the top of the list of activities, and its display can be toggled on or off by selecting **Display Summary Task** from the **View** menu (see Figure 5.7).

Figure 5.7 View Menu



In the Gantt View, supertasks are indicated by vertical cones at the end of their corresponding schedule bars.

Note that the durations of the supertasks are determined by the overall duration of their subtasks. Thus, you cannot change the duration of a supertask.

If there is no PROJECT statement, all menu selections that correspond to the multi-project structure are unavailable for selection. For example, the **Display Summary Task** selection in Figure 5.7 will appear dimmed.

Table View

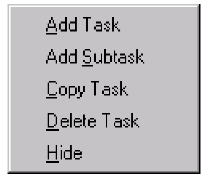
The Table View displays information about a project in tabular form. It displays activities along with their descriptions, various activity schedules, resource requirements, calendars, and target dates. The hierarchical information about an activity is provided in the Job Nbr. column by a number of small square boxes to the left of the activity number. The number of square boxes corresponds to the level of the activity in the project hierarchy. Empty boxes indicate that the activity does not have any subtasks (it is a leaf activity), while filled boxes indicate that the activity is a supertask. Some columns in the Table View are editable while others are write-protected. The editable columns are lighter in color than the noneditable ones. In general, you can type into all columns that provide input to the project, while all other columns that contain output values from PROC PM are write-protected. For example, in Figure 5.2, the WBS Code column cannot be edited, while the activity and Duration columns can be edited.

In the Table View, you can add or delete activities, add subtasks, change the order of the columns or the activities, edit activity information, and so on. These tasks are described in the following sections.

Add/Copy/Delete Tasks

Right-clicking any task in the Table View displays the pop-up menu shown in Figure 5.8. From this pop-up menu, you can Add/Copy/Delete the selected task. If you select the **Add Task** menu item, the new task is added immediately following the selected task. You can also add a subtask to the selected task by selecting the **Add Subtask** menu item. If you select the **Copy Task** menu item, a copy of the selected task is added to the bottom of the Table View. The new task has the same duration and calendar as the selected task. If the selected task is a supertask, all its subtasks (and any internal precedence constraints) are also copied.

Figure 5.8 Table View Pop-up Menu



Change Column Width

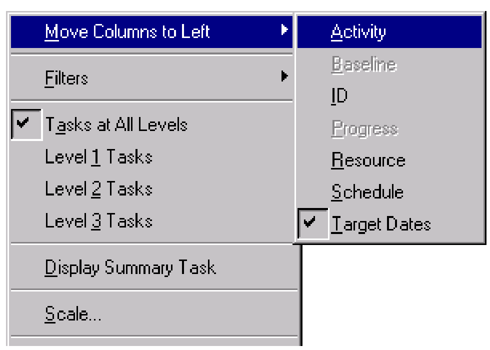
The width of a column in the Table View can be increased or decreased by dragging (with the left mouse button) the column dividers in the Table Header region of the Table View. When the pointer is positioned on the column divider, it changes to a double arrow. Dragging it to the right or left increases or decreases the width of the column.

Change the Order of the Columns

The display order of columns in the Table View can be changed in several ways:

- Drag the column in the header row to the destination.
- Select **View** from the menu and then select **Move Columns to Left** (see Figure 5.9). Choosing any of the available options moves the corresponding columns to the leftmost portion of the Table View.

Figure 5.9 Move Columns Menu



Edit Durations

To change the duration of an activity, edit the Duration column in the Table View. Note that changing an activity's duration to 0 changes the activity into a Milestone. Activity durations can also be changed in the Gantt View.

Edit Alignment Constraints

Scroll to columns named Target Date and Target Type. Enter one of the values SGE, SLE, MS, MF, FGE, or FLE in the Target Type column. You can either type the values or select them from the pop-up menu displayed by right-clicking the Target Type column (see Figure 5.10). Enter the appropriate date in the Target

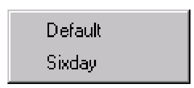
Date column. You can also view these columns by selecting **View ►Move Columns to Left ►Target Dates** from the menu (Figure 5.9). You can also change an activity's alignment constraints in the Gantt View.

Figure 5.10 Target Type Pop-up Menu

Edit Calendars

To change an activity's calendar, you can enter the calendar number in the Activity Calendar column or the calendar description in the Calendar Name column. The calendars that can be assigned to an activity are predefined in the Calendar data set. To see a list of the calendars, you can right-click in one of the calendar columns. This will pop up a list of calendars, from which you can select the activity's calendar. See Figure 5.11 for an example of a calendar pop-up menu with two calendars.

Figure 5.11 Calendar Pop-up Menu



Edit Resource Requirements

You can change the amount of resources required for an activity by editing the Resource columns in the Table View. Changing the resource requirement causes the project to be rescheduled using the new resource specifications. See "Edit Resource Requirements" on page 324 for more details.

Edit Progress Information

You can edit the actual start, actual finish, percent complete, or remaining duration for an activity by editing one of the Progress Information columns in the Table View. Note that by changing one of these columns, all the other related progress columns might also be affected. For example, entering 100 in the Percent Complete column for an activity that is in progress updates the Remaining Duration column to 0, and the Actual Finish column is filled in appropriately. You can also modify the progress information of an activity in the Gantt View.

Expand/Collapse Supertasks

Double-clicking on a supertask in the Table View toggles the expand/collapse switch. This action enables you to either view or hide all the subtasks of the supertask.

Hide Tasks

An individual task can be hidden by right-clicking the task in the Table View and selecting **Hide** from the menu shown in Figure 5.8. Tasks can also be hidden from view using several filters described in the section "Setting Activity Filters" on page 324.

Move Tasks

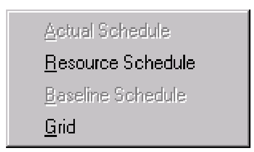
Starting anywhere in the row corresponding to an activity you want to move, drag it to the destination.

Gantt View

In the Gantt View, activity schedules are depicted by horizontal bars. There is one bar for each of the early, resource, actual, and baseline schedules. For the Early Schedule bar, critical activities are marked in different colors from the noncritical activities. Weekends are marked by shaded vertical rectangles running through the chart. Supertasks are differentiated from leaf activities by anchoring vertical cones at the ends of their Early Schedule bars.

The Gantt View is displayed with a rectangular grid that can be turned on or off by selecting **Grid** from the pop-up menu (see Figure 5.12) that is displayed by right-clicking anywhere outside of the schedule bars in the Gantt View.

Figure 5.12 Gantt View Pop-up Menu

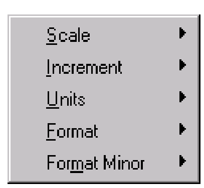


The pop-up menu in the Gantt View also enables you to toggle the display of the Actual, Resource, or Baseline Schedule bars. Note that these bars can be displayed in the Gantt View only if the project data contain the actual, resource-constrained, or baseline schedules, respectively.

In addition to displaying the activity schedules in an easy-to-view format, the Gantt View in the PM window can also be used to change the durations of the activities, add or delete precedence constraints, set activity alignment constraints, set progress information, and provide access to calendar, precedence, and resource information.

You can also change several of the display attributes of the Gantt View by using the **Time Axis** pop-up menu (see Figure 5.13) to set the scale of the axis, format the time axis labels, set the units of display, and so forth. All of these tasks are described in the following sections.

Figure 5.13 Time Axis Pop-up Menu

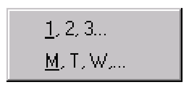


Change the Format of the Time Axis

The format of the major axis can be changed by right-clicking on the header row of the Gantt View and selecting **Format** for the major axis. For the minor axis, select **Format Minor.** Some example selections available for the formats are shown in Figure 5.14 and Figure 5.15. In addition to the formats explicitly listed for the major axis, you can specify any valid numeric format by selecting **Other** and filling in the appropriate fields in the dialog box that is opened as a result.

Figure 5.14 Major Axis Format Pop-up Menu

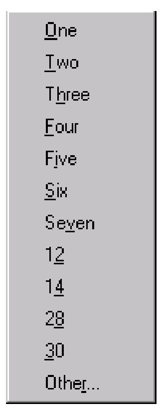
Figure 5.15 Minor Axis Format Pop-up Menu



Change Increments

Increments in the Gantt View define the number of tick marks on the minor axis per tick mark on the major axis. They can be changed by right-clicking on the header area and selecting **Increment** from the pop-up menu. The available selections are shown in Figure 5.16.

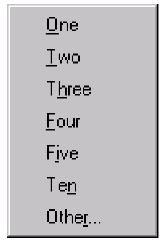
Figure 5.16 Increment Pop-up Menu



Change the Scale of the Time Axis

Move the pointer to a tick mark on the major axis in Gantt View. The pointer changes to a double arrow. Drag the tick mark horizontally to change the scale. You can also change the scale by using the **Scale** pop-up menu (see Figure 5.17) from the **Time Axis** pop-up menu.

Figure 5.17 Scale Pop-up Menu



Change Units

The Time Axis Units in the Gantt View can be changed by right-clicking on the header bar in the Gantt View and selecting **Units** (see Figure 5.18). The default value of the units used for display is based on the specification of the INTERVAL= parameter in the invocation of the PM procedure.

Figure 5.18 Units Pop-up Menu

Display/Hide Selected Schedules

A display/hide switch for a given schedule can be toggled by right-clicking in the main panel of the Gantt View and selecting the desired schedule (see Figure 5.19).

Figure 5.19 Gantt View Pop-up Menu

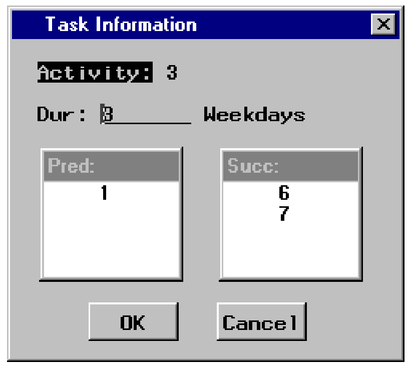
Display Task Information

You can display detailed information for an activity by right-clicking any of its schedule bars and selecting **Task Information** from the resulting pop-up menu (see Figure 5.20).

Figure 5.20 Schedule Bar Pop-up Menu

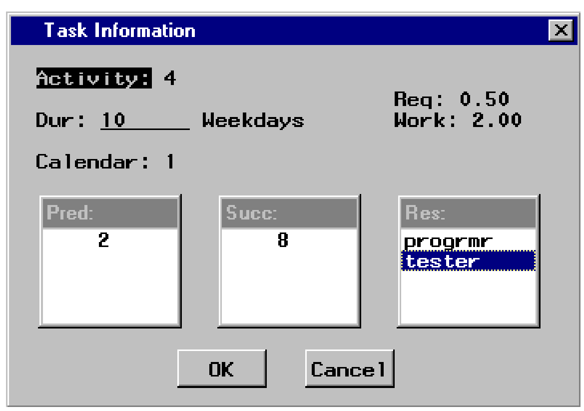
The ensuing **Task Information** dialog box (see Figure 5.21) displays the job number, duration, duration units, a list of predecessor activities, and a list of successor activities, as well as applicable calendar and resource information for the selected activity. You can also edit the activity duration from the **Task Information** dialog box.

Figure 5.21 Task Information Dialog Box



If any calendars have been defined to the project, the activity calendar is also displayed. If the project utilizes any resources, there is a list box that lists the resources required by the activity (see Figure 5.22). Selecting a resource from this list box displays the quantity required by the activity in the **Req** field. Furthermore, if the selected resource drives the duration of the activity, then the appropriate work value is also displayed in the **Work** field.

Figure 5.22 Task Information Dialog Box (Calendar and Resources)



Modify Activity Alignment Constraints

An activity's Early Schedule bar can be moved using the left mouse button. When the pointer is positioned over the activity bar, it changes to a cross-hair type. You can then drag the bar horizontally to a new position. This sets an alignment constraint of type 'SGE' for the selected activity with the align date corresponding to the one at the left edge of the bar's new position. Other types of alignment constraints can be entered by editing the Target Date and Target Type columns as described in "Edit Alignment Constraints" on page 313.

Modify Durations

You can modify the duration of an activity in several ways. In the Gantt View, you can enter it directly by using the **Task Information** dialog box as described in "Display Task Information" on page 317, or change it indirectly by altering the width of the schedule bar. To change the duration of an activity, point to the right edge of the activity's Early Schedule bar, and drag to the left or the right depending on whether you want to decrease or increase the duration. You can also edit activity durations in the Table View.

Modify Precedence Information

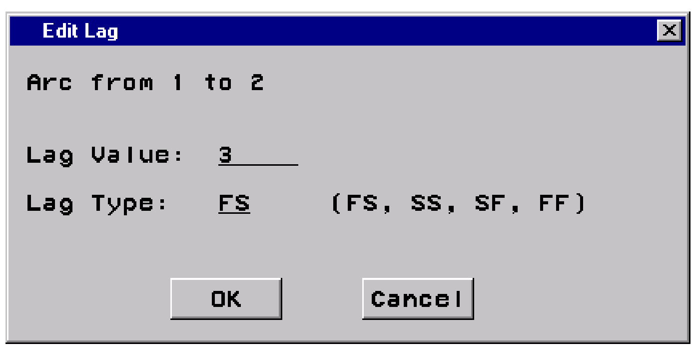
You can add precedence constraints by depressing the left mouse button at either end of the predecessor activity bar and releasing it at either end of the successor activity bar. The type of constraint (FS, FF, SS, or SF) depends on which end of the bars the constraints are drawn from.

You can delete precedence constraints by right-clicking on the arc and selecting **Delete** (see Figure 5.23).

Figure 5.23 Arc Pop-up Menu

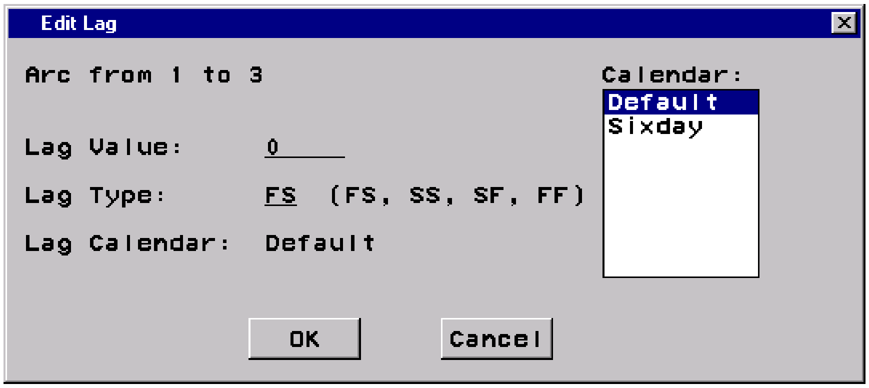
You can modify the type of the precedence constraint or the lag value associated with the precedence constraint by right-clicking on the arc and selecting **Edit Lag**. The ensuing dialog box is shown in Figure 5.24. Enter the value of the lag duration in the first field and the type of the lag in the second field. Valid values of lag are Finish-to-Start (FS), Start-to-Start (SS), Start-to-Finish (SF), and Finish-to-Finish (FF).

Figure 5.24 Edit Lag Dialog Box



If calendars are defined in the project, the **Edit Lag** dialog box includes the lag calendar associated with the selected precedence constraint. You can change the lag calendar by selecting from the list of available calendars that is displayed within the **Edit Lag** dialog box shown in Figure 5.25.

Figure 5.25 Edit Lag Dialog Box (Multiple Calendars)



Modify Progress Information

To modify the Progress using the Gantt View, you must include the Actual Schedule in the view. You can drag the actual schedule bar for the activity to change the amount of progress on the activity; you can also move the activity's actual bar to change the Actual Start of the activity.

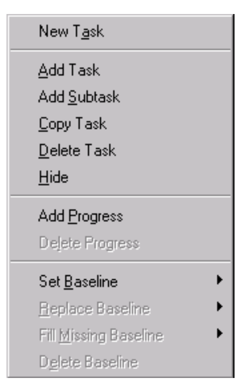
When the project contains progress information, a Timenow line is drawn in the Gantt View, indicating the TIMENOW date. You can move the Timenow line by dragging it. When you change the value of TIMENOW, the progress information changes for all the activities. A confirmation window requires you to confirm that you do want to change the progress information for all the activities. (See also "Macro Variable TIMENOW" on page 328.)

You can also edit the progress columns in the Table View.

Creating and Editing Projects

The PM window provides an easy-to-use interface to enter basic project information such as a list of activities, their durations, order of precedence, resource requirements, and so forth. You can also use the **Edit** menu (see Figure 5.26) to add or delete progress, baseline, and other information. These functions are described in the following sections.

Figure 5.26 Edit Menu



Add Activities

An activity (or task) can be added to the Project in the PM window by right-clicking in the Table View. If **Add Task** is selected from the pop-up menu, then an activity is added at the same level as the selected activity. Subtasks of an activity can be added by selecting **Add Subtask**. These actions are also available from the **Edit** menu (Figure 5.26) whenever an activity is selected in the Table View. Note that the selected activity is highlighted.

To add a new task at the topmost level of the project hierarchy, select **New Task** from the **Edit** menu.

Add Precedence Constraints

To add precedence constraints in the Gantt view, point at the right edge of the predecessor activity until the pointer changes to a cross-hair and drag it vertically up or down to the left edge of the successor activity. By starting and dropping at different ends of the activity bar, you can create nonstandard precedence relationships between the activities. You can view the predecessor and successor tasks for an activity from the **Task Information** dialog box.

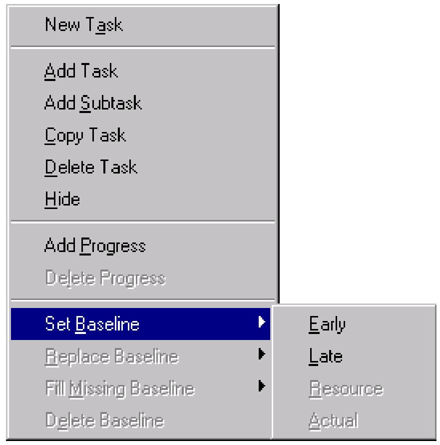
Add Baseline Information

Baseline information is saved in a project so that the current status of a project can be measured against some base schedule. The baseline information can be set in several different ways; most of the actions relating to the Baseline schedule can be performed using the selections available from the **Edit** menu (see Figure 5.26).

- If the project data include a Baseline schedule, saved in the variables B_START and B_FINISH, the PM window displays the Baseline schedule when it is first invoked. This schedule can be replaced by selecting **Replace Baseline** from the **Edit** menu. This selection can be used to reset the Baseline schedule to a new schedule corresponding to one of the current schedules.

- If the project data do not include a Baseline schedule, the Baseline schedule can be set in the PM window by selecting **Set Baseline** from the **Edit** pull-down menu (see Example 5.3). This selection can be used to set the Baseline schedule to one of the current schedules (see Figure 5.27). Thus, selecting **Resource** from the **Set Baseline** menu sets the baseline schedule to the current resource-constrained schedule. By saving the current resource-constrained schedule, you can perform some what-if analysis by changing some of the resource requirements or other parameters of the project and comparing the resulting schedule with the saved baseline schedule.

Figure 5.27 Set Baseline Pull-down Menu



- The individual Baseline values can also be edited in the Table View by changing the values in the Baseline Start and Baseline Finish columns.
- If new activities are added to the project, the Baseline values for the new tasks are missing. These can be set to correspond to the current schedule values by selecting **Fill Missing Baseline** from the **Edit** menu.
- If you want to delete the Baseline information from the project data, you can select **Delete Baseline** from the **Edit** menu.

Add Progress Information

Progress information can be included by using the ACTUAL statement, which is similar to the one for PROC CPM. If the PM window is invoked without the ACTUAL statement, then progress information can be added to the Project from the **Edit** menu (Figure 5.26) by selecting **Add Progress**.

Progress information is updated by dragging the actual schedule bars horizontally (in a manner similar to the one for changing durations) in the Gantt View or by modifying the values in the Progress columns in the Table View. See “Modify Progress Information” on page 320 and “Edit Progress Information” on page 314.

For details about how the progress information is used to update the project schedule, see “Progress Updating” on page 109. See also Example 5.6.

Change Duration

The duration of an activity can be changed directly from the Duration column of the Table View or the **Task Information** dialog box of the Gantt View. It can also be changed indirectly by dragging the activity bar at the right edge in the Gantt View.

Copy Activities

An activity (or task) can be copied in the PM window by right-clicking in the Table View. If **Copy Task** is selected from the pop-up menu, then a copy of the selected activity is added at the end of the activities listed in the Table View. The new task has the same duration and calendar as the selected task. If the selected task is a supertask, all its subtasks (and any internal precedence constraints) are also copied.

Create Milestones

You can create milestones by adding an activity and assigning it zero duration.

Create Nonstandard Precedence Relationships

A Finish-to-Start relationship between two activities is considered to be a standard precedence constraint. You can create it in Gantt view by dragging the precedence constraint from the right end of the predecessor activity bar to the left end of the successor activity bar. Nonstandard precedence constraints are created by starting and ending at different ends of the two activity bars. For example, a Start-to-Finish relationship is created by dragging from the left end of the predecessor activity bar to the right end of the successor activity bar.

In addition to specifying the type of the precedence constraint, you can also specify a lag or lead time between the two activities. This lag value can be edited from the Gantt View. See "Modify Precedence Information" on page 319 for more details.

Create Subtasks

Subtasks can be created only if the PM procedure is invoked with the PROJECT statement. To create a subtask, right-click on the parent activity in the Table View. Then select **Add Subtask** from the background menu. The newly created subtask has one more little square box than the parent task in the Job Nbr. column in the Table View. The empty square boxes denote that it is a leaf activity (a task with no subtasks). The number of boxes denote a task's level in the project hierarchy, starting with level 0 for the Project Summary task.

Delete Activities

An activity can be deleted in the Table View by right-clicking anywhere in the task row and selecting **Delete Task**. If the selected task is a supertask, all its subtasks are also deleted. Note that, in this case, a confirmation dialog box confirms the **Delete Supertask** action.

Delete Precedence Constraints

To delete a precedence constraint, right-click anywhere on the arc and select **Delete** from the pop-up menu. You can view the predecessor and successor tasks for an activity from the task information window.

Edit Activity Alignment Constraints

Activity alignment constraints can be added/modified as described in "Edit Alignment Constraints" on page 313and "Modify Activity Alignment Constraints" on page 318, respectively.

Edit Baseline Information

To edit the baseline schedule, scroll to the Baseline Start and Baseline Finish columns and type in the new values of the baseline start and finish times. Note that you cannot change the baseline values by moving the Baseline Schedule bars. See "Add Baseline Information" on page 321 for more details.

Edit Calendar Specifications

Calendars are defined by the CALEDATA= option in the PROC PM statement. This option is similar to the corresponding option in PROC CPM. After calendars are defined in the Project, an activity's calendar can be changed or set in the Table View by editing the Activity Calendar or Calendar Name columns. You can either type the values or select them from the menu displayed by right-clicking in either of the Calendar columns. See "Edit Calendars" on page 314. You can also view the activity calendar from the task information window.

Edit Resource Requirements

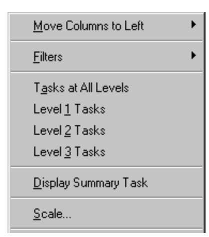
The resource requirement information for each activity is displayed and can be edited in the Table View. A column for a resource is created in the Table View when it is specified in the RESOURCE statement of the PROC PM invocation. For details about the RESOURCE statement, the Resource data set, and Resource Allocation, see Chapter 4, "The CPM Procedure." Changing the resource requirement causes the project to be rescheduled to use the new resources. You can also view the resource requirements for an activity from the task information window.

If alternate resources are used by the scheduling algorithm, an extra set of columns is added to the Table View. These columns (one for every resource in the project) display the resources that were actually used. These Usage columns for the resources cannot be edited.

Setting Activity Filters

Activity filters can be set by using the project hierarchy or by selecting from a list of activity attributes, as described in this section.

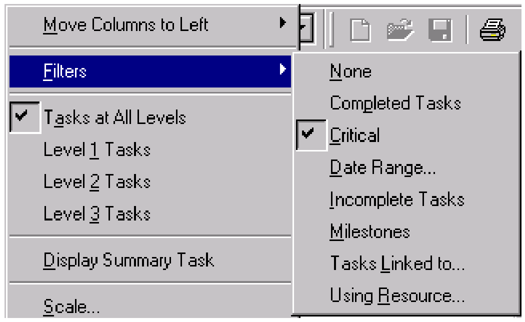
Figure 5.28 View Menu



Activities at different levels in the hierarchy can be viewed by selecting **View** from the menu (Figure 5.28) and selecting the appropriate level of the project hierarchy to filter out the higher level tasks. For example, selecting **Level 2 Tasks** displays only the tasks that are at Level 2 or lower. All activities can be viewed by selecting **Tasks at All Levels** from the **View** menu.

Activities can also be filtered using different criteria by selecting **View ►Filters** from the **View** menu (see Figure 5.28). The available filters are shown in Figure 5.29. By default, no filter is in effect (the selection is **None**); you can save the filter of your choice in the Preference data set (see "Saving and Restoring Preferences" on page 325).

Figure 5.29 Filters Pull-down Menu



Saving and Restoring Preferences

When the PM window is displayed for the first time for a given project, the order and width of the columns in the Table View, the font used for the display, the size of the window, the boundary between the Table and the Gantt Views, and several other attributes of the display are determined by the procedure. As you add activities and edit the Table View, you can change some of these attributes according to your preference. You can also select a different level of display or set some activity filters (see "Setting Activity Filters" on page 324).

PROJECT Data Set

PROC PM enables you to save the attributes of the display in an indexed data set that is specified in the PROC PM statement by using the PROJECT= option. The following preferences can be saved from one invocation to another:

- text font
- time increment
- time units
- major time axis format
- minor time axis format
- schedule bars displayed (for example, Actual, Baseline, and so forth)
- chart grid
- chart scale
- table column widths
- table column order
- Table View-Gantt View dividing line
- activity filters

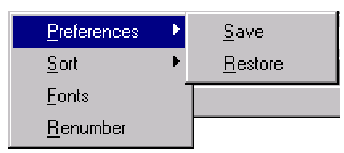
- activity level
- project summary
- window dimensions

The Project data set uses three variables to save the preference information:

- PROJATTR—contains a keyword identifying the project attribute. Each attribute has either a numeric value or a character value. The length of this variable is 8.
- PRATNVAL—used for numeric data corresponding to the attribute.
- PRATCVAL—used for character data corresponding to the attribute. The length of this variable is 200.

You can save and restore the preferences from the **Project** menu, which contains the **Preferences** submenu (Figure 5.30). Note that you have to explicitly save the project preferences using the **Save** selection from this menu. Closing the PM window saves only the activity data of the project; it does not automatically save the project preferences. When you restore preferences, the state used is the one that was last saved for the project in the specified preference data set.

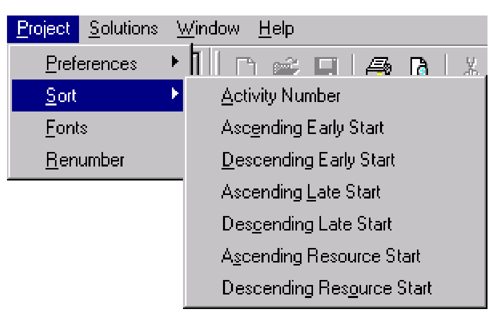
Figure 5.30 Preferences Menu



Sorting Activities

Activities can be sorted by activity number, early start, late start, and resource start by selecting **Project ►Sort** from the **Project** pull-down menu (see Figure 5.31). Once the activities are sorted, the Schedule output data set contains the activities in the new sorted order. See "Renumbering the Activities" on page 327.

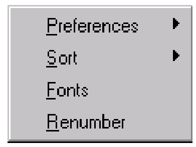
Figure 5.31 Sort Menu



Setting the Project Font

When the PM window is first displayed, the font used in all the text areas of the window is the same as the SAS font used in other windows. You can use the **Fonts** selection in the **Project** menu (Figure 5.32) to change the font used in the PM window. You can select the various fonts and their sizes from the font manager thus obtained. This font can also be saved (and restored) in the Project data set.

Figure 5.32 Project Menu



Renumbering the Activities

When the PM window is first displayed for the specified project, the activities are listed in the order in which they are defined in the Activity data set. The activity numbers displayed in the Job Nbr. column correspond to this same order. Even if the activities are rearranged, either by moving selected activities or by sorting, these numbers do not change. Likewise, no renumbering takes place automatically if activities are deleted from the project.

You can use the **Renumber** selection in the **Project** menu (Figure 5.32) to reassign consecutive numbers to the activities, starting from the first activity displayed.

When you close the PM window, saving all the activity information to the Schedule data set, the activities are numbered according to the order in which they were displayed at the end of the editing session. In other words, the **Close** action implicitly invokes the **Renumber** command on the project activities. These activity numbers are, in fact, saved as the values of the ACTID variable (see "Schedule Data Set" on page 329).

Printing

The PM window provides functionality to print the Gantt View, the Table View, or both, provided that a printer has been selected and the correct information has been set in the Printer Setup window. **Print Preview** can be used to view the information before printing, and the printed output can be saved to a file. All the printing functions are available from the **File** menu (Figure 5.33).

Figure 5.33 File Menu

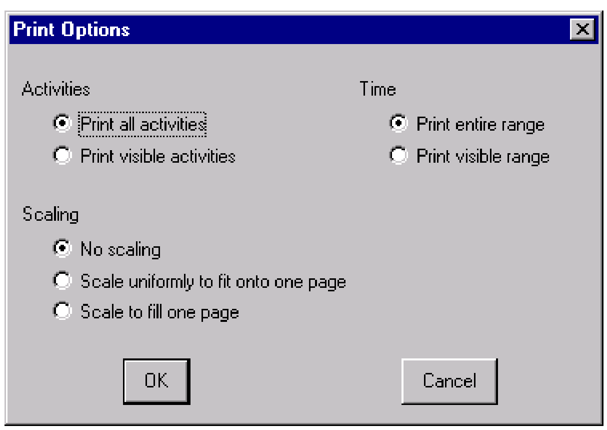
Preview the Printed Output on the Screen

You can view the printed output on screen before actually printing it by selecting **Print Preview** from the **File** menu.

Print Options

Select **Print Options** from the **File** menu. There are options for selecting time and activity axis range and scaling of the printed output. See Figure 5.34.

Figure 5.34 Print Options Dialog Box



Save the Printed Output to a File

The printed output can be saved to a file by selecting **File ►Print ►Print to File**.

Macro Variable TIMENOW

The PM window can be used to add and edit progress information to a project. When progress information is added, the Schedule data set contains all the progress variables; see "Progress Updating" on page 109.

However, all the values of the progress variables are reconciled and revised on the basis of the value of the TIMENOW parameter. Since the PM procedure enables you to move the TIMENOW line as well as implicitly change the value of TIMENOW by updating the Actual Start or Finish times of the activities, the value of TIMENOW at the end of the editing session is an important parameter of the project. This value is saved in a macro variable called TIMENOW and can be used in subsequent editing sessions of the same project. See Example 5.6 for an example of the use of the TIMENOW macro variable.

Summary of Differences

The computation of the schedule, the resource-constrained scheduling algorithm, the resource usage information, and all other aspects of the scheduling engine for PROC PM are the same as the ones for PROC CPM. Refer to Chapter 4, "The CPM Procedure," for details. Some minor differences that pertain to the Schedule Data set and ALIGNTYPE statement are explained in the following sections.

Schedule Data Set

The Schedule data set produced by PROC PM is very similar to the Schedule data set produced by PROC CPM. See "OUT= Schedule Data Set" on page 99.

However, unlike PROC CPM, the PM procedure is interactive in nature; it enables you to add activities, set precedence constraints, reorder the activities, and so on. Thus, the output data set produced by PROC PM is designed to capture the original project data as well as all the changes that are made to the project in the course of the interactive session.

There are several differences between the forms of the Schedule output data sets produced by the PM and CPM procedures:

- The PM procedure automatically includes all relevant variables that are needed to define the project. Thus, the ACTIVITY, SUCCESSOR, LAG, DURATION, ALIGNDATE, and ALIGNTYPE variables are included in the output data set by default. If the RESOURCE statement is used, all the resource variables are also included. Likewise, if actual progress is entered for the project during the course of the interactive session, all the progress-related variables are added to the output data set.

- The PM procedure contains three sets of observations, identified by three different values of a new variable, OBS_TYPE. The first set of observations contains one observation for every activity in the project. The value of the OBS_TYPE variable for these observations is `'SCHEDULE.'` These observations contain all the activity information such as the duration, the start and finish times and the resource requirements. The second set of observations contains one observation for every precedence constraint in the project. The value of the OBS_TYPE variable for these observations is `'LOGIC.'` These observations contain all the precedence information such as the activity, successor, and lag information.

  The third set of observations is present only if the project has resource-driven durations. The value of the OBS_TYPE variable for these observations is `'WORK.'` These observations specify the WORK value for each resource used by each activity in the project.

- The order of the activities in the Schedule data set produced by PROC PM corresponds to the order in which the activities appear in the Table View at the end of the interactive session. Likewise, when the procedure is first invoked, the order of the activities in the Table View corresponds to the order in which the activities are defined in the Activity input data set. If, during the course of the session, some of the activities are reordered or deleted, or if some new activities are added, the Schedule output data set contains all the activities that are defined in the Table View at the end of the session.

- The PM procedure also assigns a numeric identifier for each activity. These values are assigned by PROC PM consecutively in the order of the activities in the Table View and are saved in a variable called ACTID (see "Renumbering the Activities" on page 327). In addition to the ACTID variable, the Schedule data set also contains a numeric variable called SUCCID, which contains the numeric identifier for the successor activities in the observations for which OBS_TYPE='LOGIC.' If the PROJECT statement is used in the invocation of the PM procedure, a numeric variable called PNTID is added to the Schedule data set; this variable identifies the parent task for each activity.

  NOTE: If the ACTIVITY variable in the Activity input data set is a character variable, the ACTID, SUCCID, and PNTID variables are added to the Schedule data set in addition to the ACTIVITY, SUCCESSOR, and PROJECT variables. On the other hand, if the ACTIVITY variable in the Activity input data set is numeric, the new ACTID, SUCCID, and PNTID variables replace the numeric ACTIVITY, SUCCESSOR, and PROJECT variables, respectively.

ALIGNTYPE Statement

In PROC PM, if an ALIGNTYPE variable is specified but no ALIGNDATE variable is specified, then no error message is generated; PROC PM ignores the ALIGNTYPE variable and generates a schedule. However, in PROC CPM, this results in an error message with no schedule generated.

RESOURCE Statement

In PROC CPM, the NORESOURCEVARS option in the RESOURCE statement requests that the variables specified in the RESOURCE statement be dropped from the Schedule data set. However in PROC PM, this has no effect.

Examples: PM Procedure

This section illustrates some of the interactive features of PROC PM by using a few simple examples that lead you through different stages of entering and editing project data. A simple software development project is used in all the examples. The output data set from one example is used as input to the next example. Where necessary, additional data sets are created, or the input data set is modified using simple DATA step code.

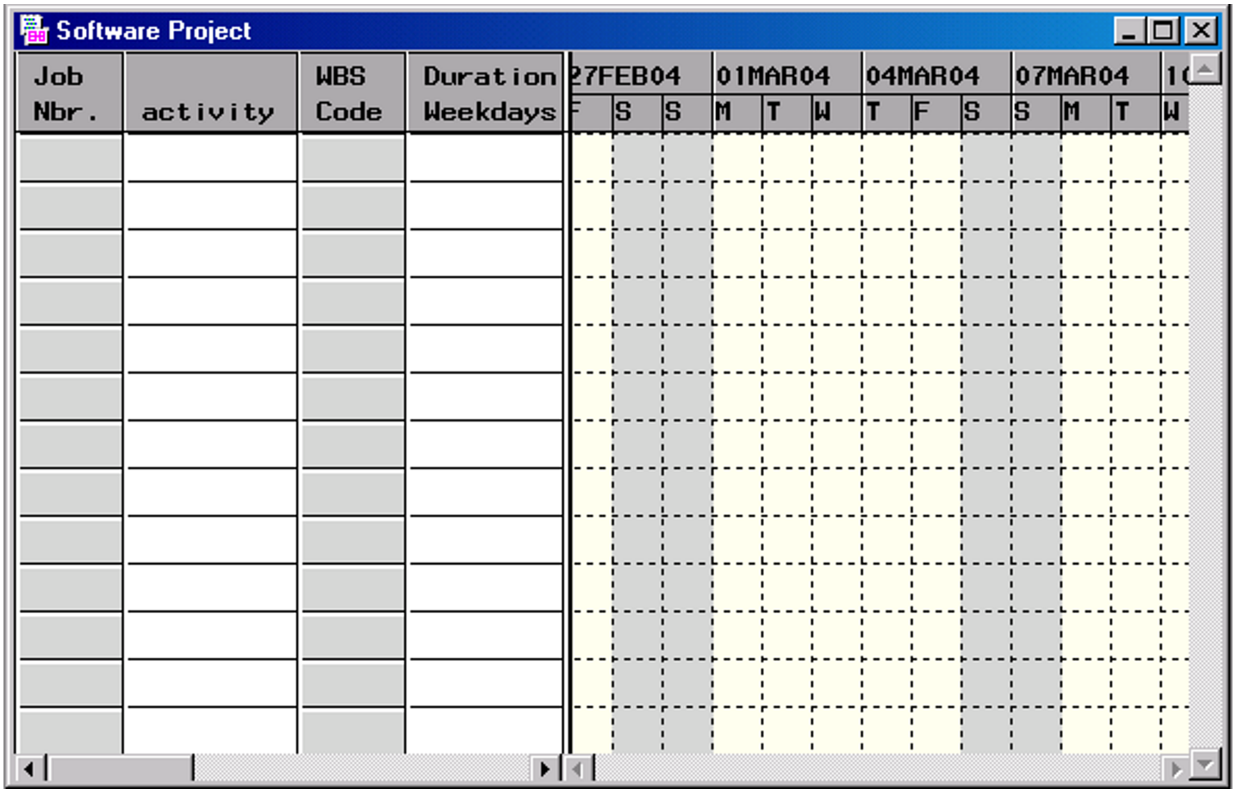
Example 5.1: Defining a New Project

In this example, a simple software development project is built from scratch, starting with an empty Activity data set. PROC PM is invoked with an Activity data set that has no observations and just a few variables that are required to start the procedure. In addition to the Activity data set, a Project data set is also defined that is used to save the display attributes of the PM window to be used between successive invocations of the procedure. The following program invokes PROC PM and opens a PM window that enables you to enter project data. The initial window is shown in Output 5.1.1.

Note that the PROJNAME= option is used in the PROC PM statement. This value is used to label the PM window. Also specified in the PROC PM statement is the PROJECT= option that identifies the project attribute data set. The activities in the project follow a weekday calendar which is indicated to PROC PM by specifying the INTERVAL=WEEKDAY option. In the PM window, the weekends are shaded gray in the Gantt View.

```
/* Initialize the Activity data set */
data software;
   length activity $20.;
   input activity $ actid succid pntid duration;
   datalines;
;

data softattr;
   length projattr $8. pratcval $200.;
   input projattr pratnval pratcval;
   datalines;
;

proc pm data=software project=softattr
        date='1mar04'd interval=weekday
        projname='Software Project'
        out=softout1;
   act actid;
   succ succid;
   project pntid;
   duration duration;
   id activity;
   run;
```

Output 5.1.1 Initial PM window

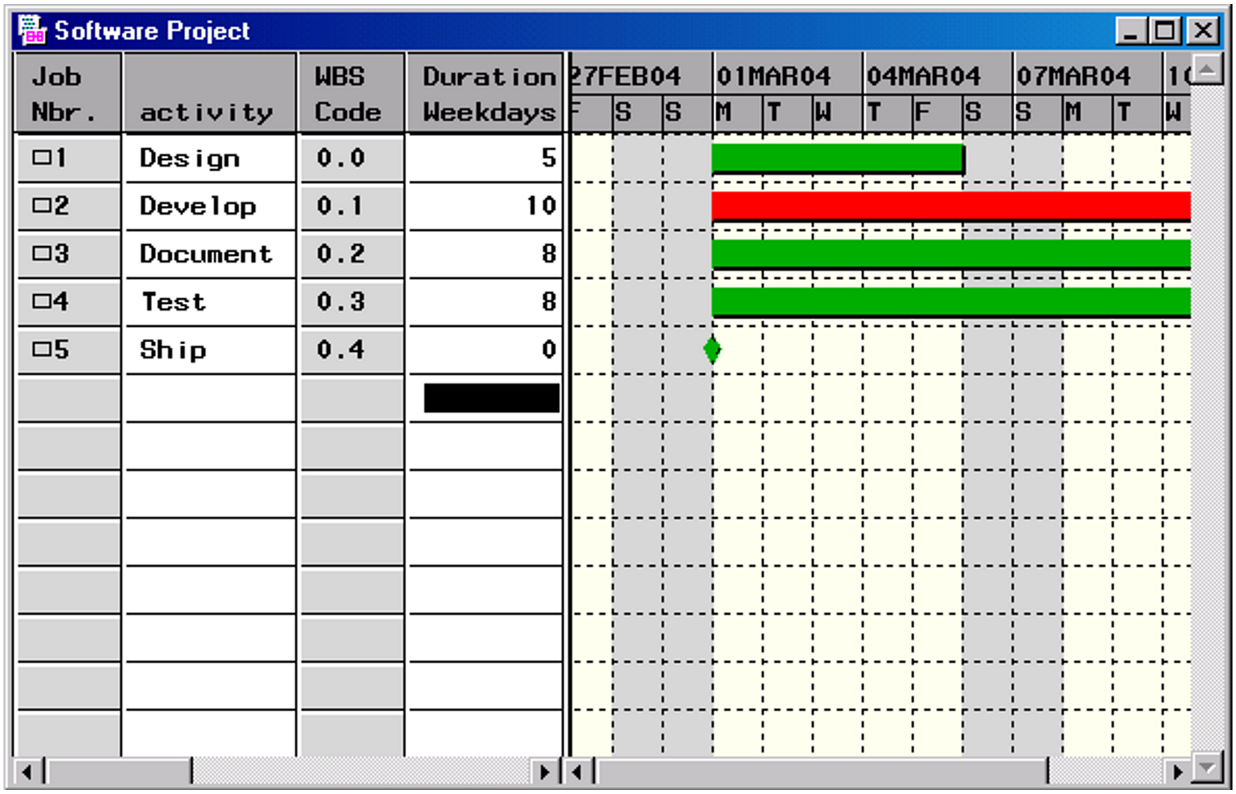


In the PM window, enter the following tasks with the corresponding durations in the Table View:

```
Design         5
Develop       10
Document       8
Test           8
Ship           0
```

As each task is entered, the Schedule columns in the Table View are updated with the early and late start times, and the Early Schedule bars appear in the Gantt View. Output 5.1.2 shows the PM window after the five tasks have been entered. To view the Schedule columns, you can scroll the Table View to the right or use the **View** menu (Figure 5.7) to move the Schedule columns to the left.

Output 5.1.2 List of Tasks in the Software Project

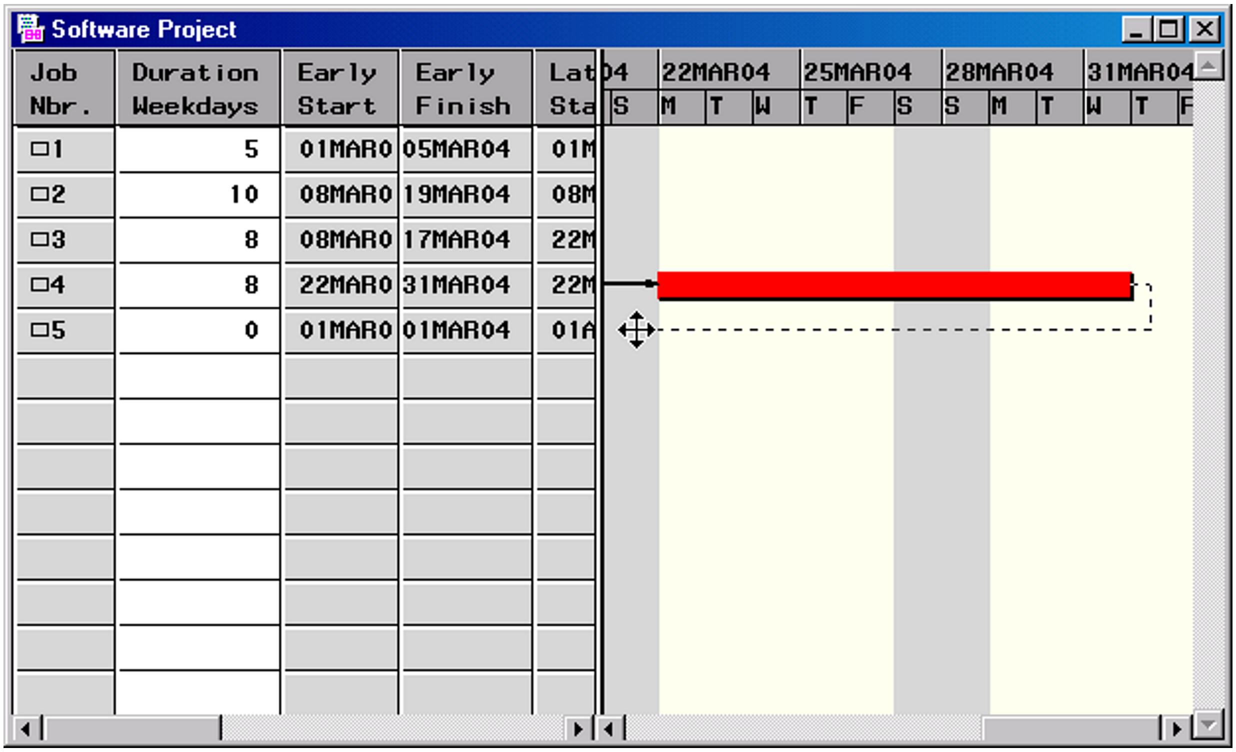


To enter precedence constraints between two activities, such as ‘Design’ and ‘Develop,’ draw an arc, using the left mouse button, from the end of the predecessor task to the beginning of the successor task. Use the Gantt View to enter the following precedence constraints:

```
Design  --> Develop
Design  --> Document
Develop --> Test
Test    --> Ship
```

Output 5.1.3 shows the Software Project as the last precedence constraint is being drawn. Note that, in this view of the PM window, the Schedule columns have been moved to the left, the grid lines in the Gantt View have been turned off (using the menu in Figure 5.12), and the Gantt View has been scrolled to the right to bring the end of the schedule bar for ‘Test’ into view.

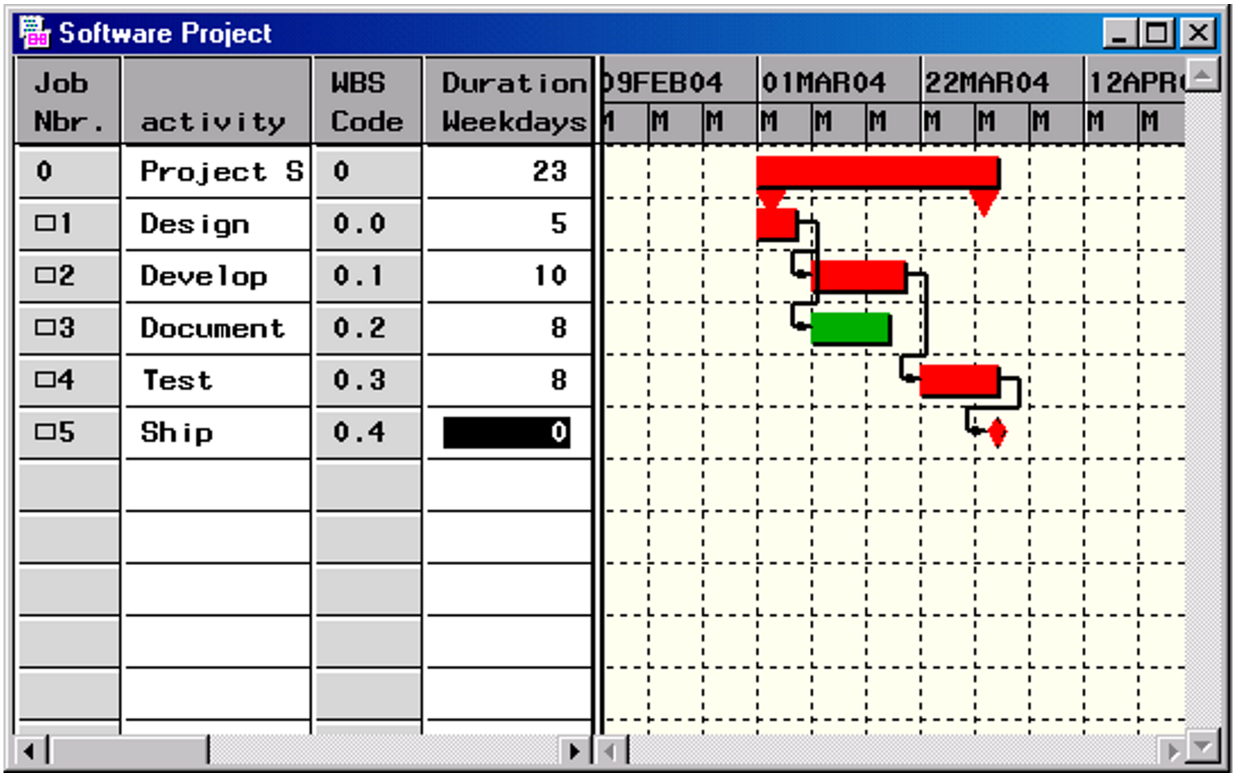
Output 5.1.3 Drawing Precedence Constraints



To check the overall project status, you can bring the Project Summary task into view by selecting **Display Summary Task** from the **View** menu (Figure 5.7). Note that the project duration is 23 days. The critical activities are shown in red while the noncritical ones are green. The Summary Task is indicated by vertical cones at the end of its schedule bar.

For the next few examples, the units used in the Gantt View are changed to "Weeks" by using the **Axis** pop-up menu shown in Figure 5.13, the Summary Task is displayed at the top of the list of activities, and the Activity description columns are shown in the Table View. To save these window settings in the Project data set, select **Project ►Preferences ►Save** from the **Project** menu. The view of the project corresponding to these settings is shown in Output 5.1.4. You can end the interactive editing session by closing the window. All the activity and precedence information is saved in the output data set, SOFTOUT1, displayed in Output 5.1.5. Note the two sets of observations in this data set: the first contains all the schedule information for all the activities, and the second lists all the precedence relationships between activities.

Output 5.1.4 Project Schedule



Output 5.1.5 Schedule Data Set

Schedule Data Set

Obs	OBS_TYPE	PNTID	PROJ_DUR	PROJ_LEV	WBS_CODE	ACTID	SUCCID	duration
1	SCHEDULE	.	23	0	0	0	.	23
2	SCHEDULE	0	.	1	0.0	1	.	5
3	SCHEDULE	0	.	1	0.1	2	.	10
4	SCHEDULE	0	.	1	0.2	3	.	8
5	SCHEDULE	0	.	1	0.3	4	.	8
6	SCHEDULE	0	.	1	0.4	5	.	0
7	LOGIC	.	.	.		1	2	5
8	LOGIC	.	.	.		1	3	5
9	LOGIC	.	.	.		2	4	10
10	LOGIC	.	.	.		4	5	8

Obs	PARENT	activity	SUCCESSR	LAG	ALGNDATE	ALGNTYPE	E_START	E_FINISH
1		Project Summary			.		01MAR04	31MAR04
2	Project Summary	Design			.		01MAR04	05MAR04
3	Project Summary	Develop			.		08MAR04	19MAR04
4	Project Summary	Document			.		08MAR04	17MAR04
5	Project Summary	Test			.		22MAR04	31MAR04
6	Project Summary	Ship			.		01APR04	01APR04
7		Design	Develop		.		.	.
8		Design	Document		.		.	.
9		Develop	Test		.		.	.
10		Test	Ship		.		.	.

Obs	L_START	L_FINISH	T_FLOAT	F_FLOAT	ES_ASC	ES_DESC	LS_ASC	LS_DESC
1	01MAR04	31MAR04	0	0	0	0	0	0
2	01MAR04	05MAR04	0	0	1	5	1	5
3	08MAR04	19MAR04	0	0	2	4	2	4
4	22MAR04	31MAR04	10	10	3	3	3	3
5	22MAR04	31MAR04	0	0	4	2	4	2
6	01APR04	01APR04	0	0	5	1	5	1
7	.	.	.	.	.	.	.	.
8	.	.	.	.	.	.	.	.
9	.	.	.	.	.	.	.	.
10	.	.	.	.	.	.	.	.

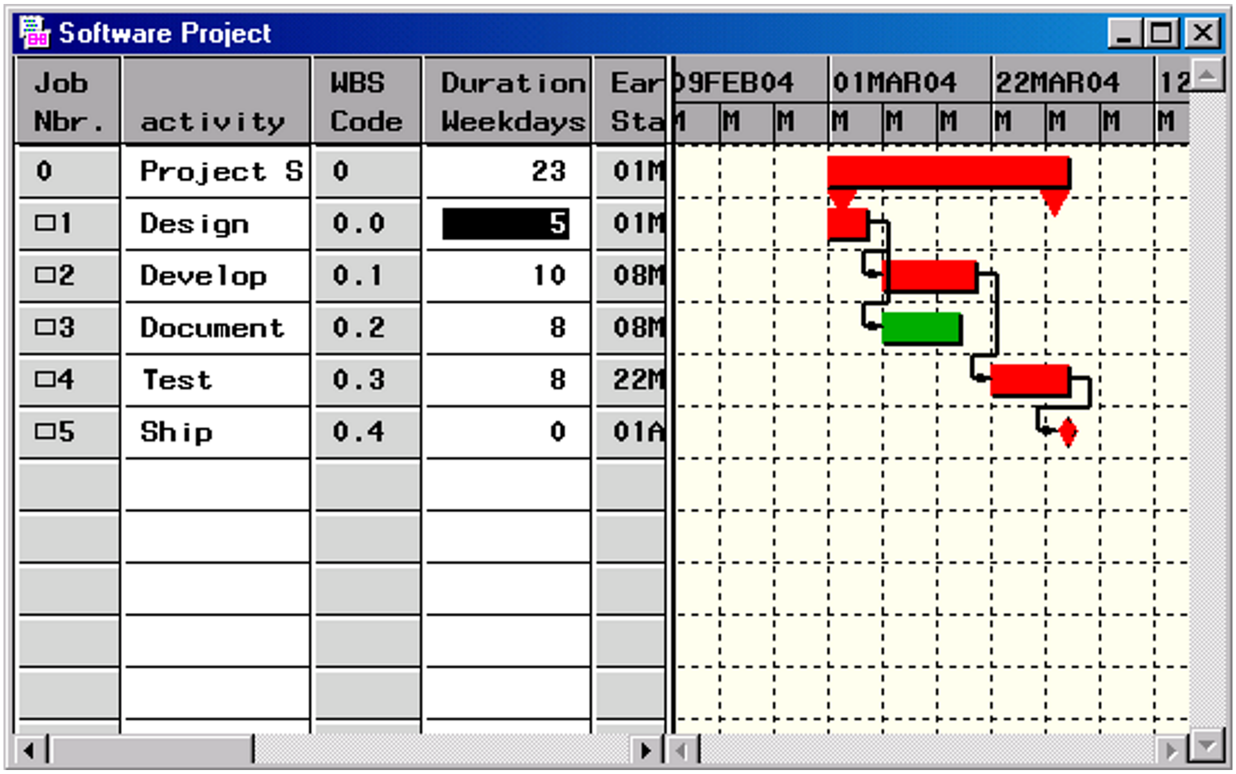
Example 5.2: Adding Subtasks to a Project

In this example, the output data set from Example 5.1 is used as input to PROC PM. The following statements bring up the saved view of the project as shown in Output 5.2.1. Note that this view is identical to the view saved in Output 5.1.4.

```
proc pm data=softout1 project=softattr
        date='1mar04'd interval=weekday
        projname='Software Project'
        out=softout1;
```

```
act actid;
succ succid;
project pntid;
duration duration;
id activity;
run;
```

Output 5.2.1 Project Schedule



In the invocation of PROC PM, the output data set name is the same as the input data set. Thus, it is possible to make changes to the Activity data set using PROC PM and then save the results back to the original data set.

In the current view of the Software Project, you want to add some subtasks to the 'Design' and 'Develop' tasks. Suppose that these two tasks are broken into two subtasks each: one for 'Module 1' and the other for 'Module 2.' Further, you want to remove the precedence constraint between the 'Design' and 'Develop' phases and add constraints between the respective modules. You can accomplish these tasks by making the following editing changes in the PM window.

1. Use the Table View pop-up menu to add the following subtasks to 'Design':

```
Module 1:    5 days
Module 2:    3 days
```

2. Add a link from 'Module 1' to 'Module 2.'

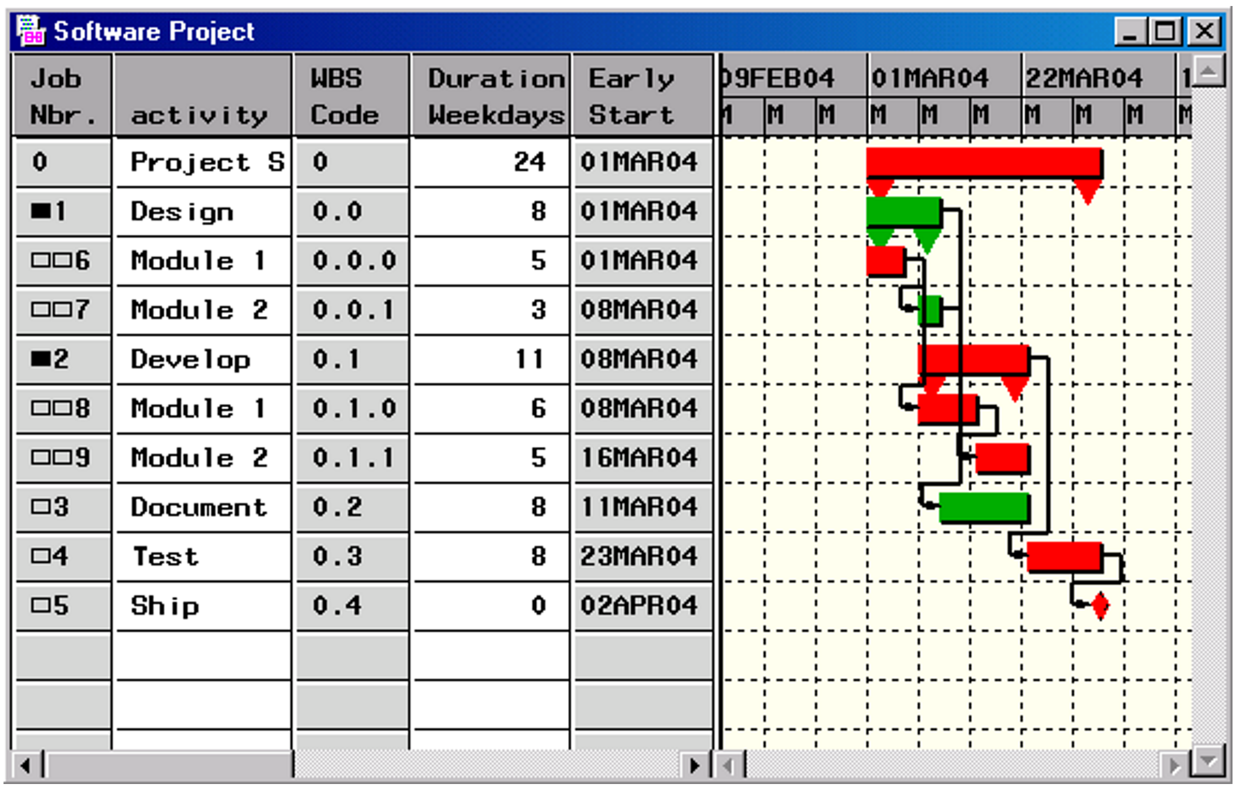
3. Use the Table View pop-up menu to add the following subtasks to 'Develop':

```
Module 1:    6 days
Module 2:    5 days
```

4. Add a link from ‘Module 1’ to ‘Module 2.’
5. Remove the link between the supertasks ‘Design’ and ‘Develop’ by clicking on the arc and selecting **Delete** from the pop-up menu.
6. Add a link from ‘Module 1’ under ‘Design’ to ‘Module 1’ under ‘Develop.’
7. Add a link from ‘Module 2’ under ‘Design’ to ‘Module 2’ under ‘Develop.’

The resulting project schedule is displayed in Output 5.2.2 and saved in the data set SOFTOUT1. Note that the new project duration is 24 days.

Output 5.2.2 Project Schedule



Job Nbr.	activity	WBS Code	Duration Weekdays	Early Start
0	Project S	0	24	01MAR04
■1	Design	0.0	8	01MAR04
□□6	Module 1	0.0.0	5	01MAR04
□□7	Module 2	0.0.1	3	08MAR04
■2	Develop	0.1	11	08MAR04
□□8	Module 1	0.1.0	6	08MAR04
□□9	Module 2	0.1.1	5	16MAR04
□3	Document	0.2	8	11MAR04
□4	Test	0.3	8	23MAR04
□5	Ship	0.4	0	02APR04

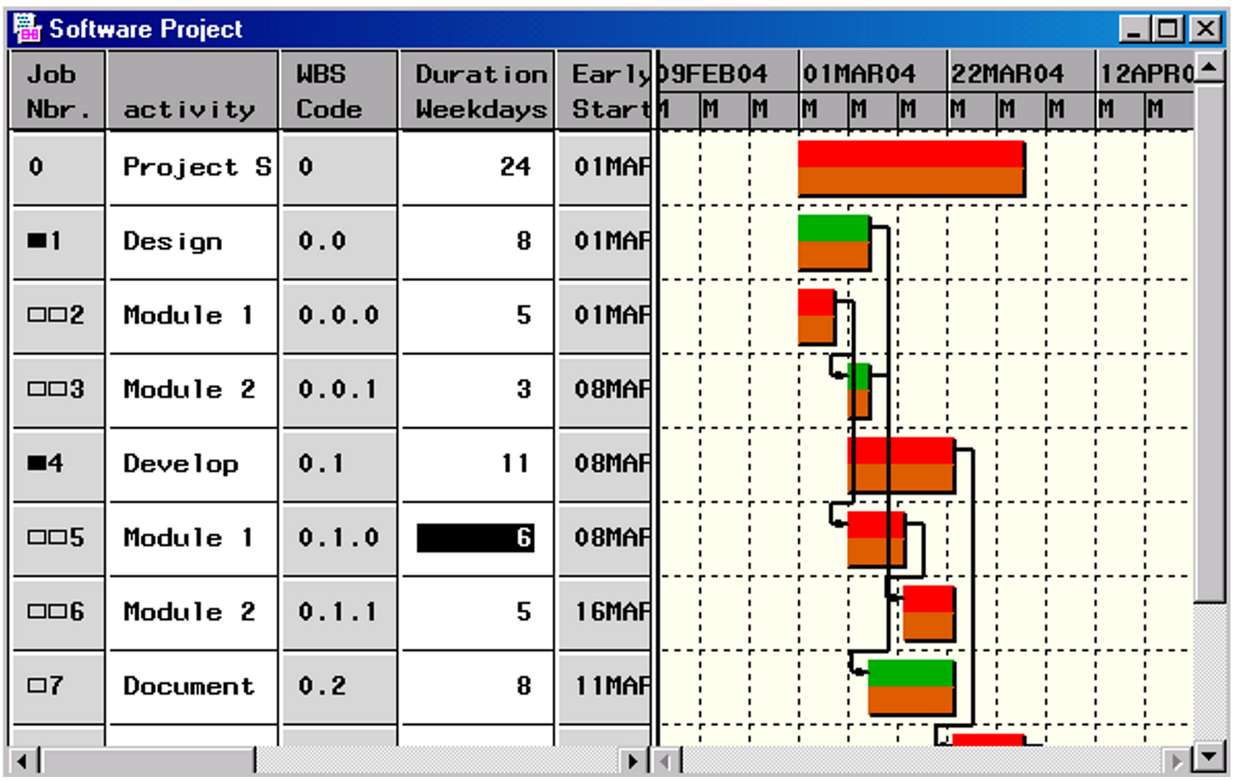
Example 5.3: Saving and Comparing Baseline Schedules

This example shows you how to save a baseline schedule and use it for comparing new schedules. Recall that in Example 5.2 the Schedule data are saved in the data set SOFTOUT1. Thus, the following invocation of PROC PM displays the Software project in its last saved state (as in Output 5.2.2, but with the WBS codes filled in). At the end of the editing session, the schedule is saved in the data set SOFTOUT3.

```
proc pm data=softout1 project=softattr
        date='1mar04'd interval=weekday
        projname='Software Project'
        out=softout3;
   act actid;
   succ succid;
   project pntid;
   duration duration;
   id activity;
   run;
```

Use the **Edit ►Set Baseline** pull-down menu (Figure 5.27) to save the current Early Schedule as a Baseline Schedule. The resulting display is shown in Output 5.3.1. Note that the Gantt View now shows the Baseline Schedule in addition to the Early Schedule. Also, the activities have been numbered to be sequential in the current view (see the section "Renumbering the Activities" on page 327).

Output 5.3.1 Using Baseline Schedules



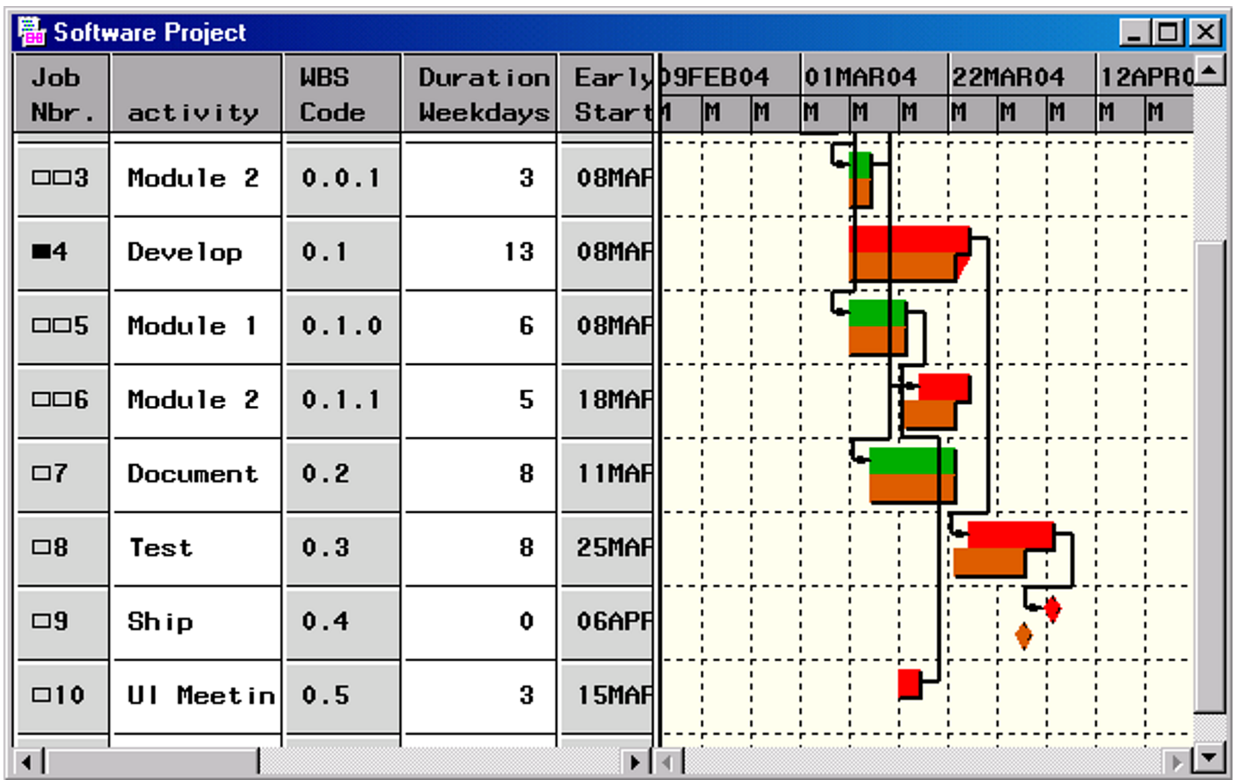
The baseline schedule is useful in determining the effect of changes to the project on the schedule. For example, suppose there is a directive from the director of your division that all the developers are required to attend a User Interface Standards Meeting before starting the development of Module 2. This meeting has been scheduled to start on March 15, 2004, and is expected to take 3 days. What is the effect of this directive on your project schedule?

To see the effect, you can make the following changes in the PM window:

1. Add a new task to the project by selecting **New Task** from the **Edit** pull-down menu.
2. To edit the newly entered task, you may need to scroll down.
3. Type in the name of the task: ‘UI Meeting.’ Set its duration to 3.
4. In the Gantt View, draw a link from this new task to Task 6 (‘Module 2’ under ‘Develop.’)
5. Also in the Gantt View, drag the task, ‘UI Meeting,’ to the tick mark corresponding to 15Mar04.

The resulting view is shown in Output 5.3.2. Note that the view may differ depending on the display parameters of your device. It is easy to see that, due to the 3-day meeting that is mandated, there is a delay in the project schedule (the project duration is now 26 days).

Output 5.3.2 Effect of UI Meeting on Schedule



You can get a complete picture of the effect on the schedule by examining all the Schedule columns that are shown in the Table View. Output 5.3.3 shows the Schedule columns, the Baseline columns, and the Target Date and Type columns in the Table View. To obtain this view, some of the columns have been moved and the Baseline Schedule bars (in the Gantt View) have been hidden from the display.

Output 5.3.3 Table View Showing All Schedules

Software Project

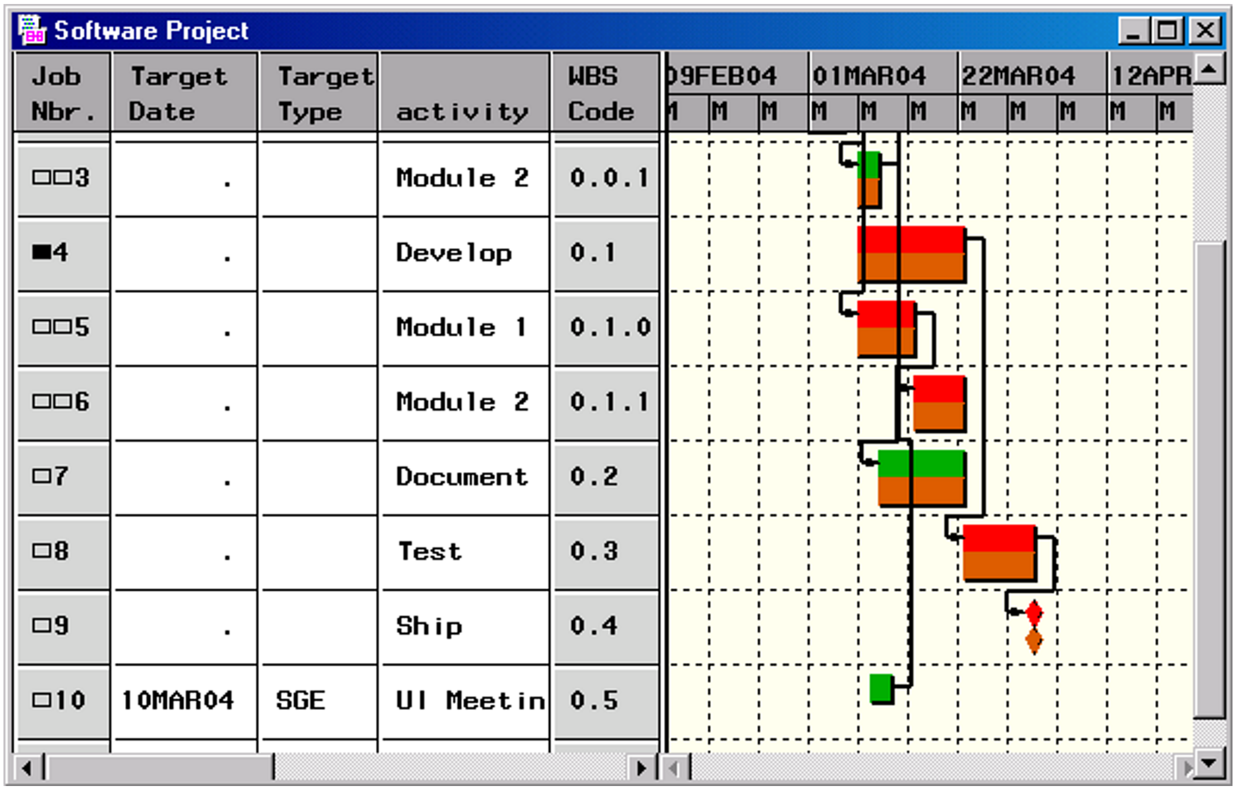
Job Nbr.	activ	Dura Week	Early Start	Early Finish	Late Start	Late Finish	Baselin Start	Baselin Finish	Target Date	Targ Type
0	Proje	26	01MAR04	05APR04	03MAR04	05APR04	01MAR04	01APR04	.	
■1	Desig	8	01MAR04	10MAR04	03MAR04	17MAR04	01MAR04	10MAR04	.	
□□2	Modul	5	01MAR04	05MAR04	03MAR04	09MAR04	01MAR04	05MAR04	.	
□□3	Modul	3	08MAR04	10MAR04	15MAR04	17MAR04	08MAR04	10MAR04	.	
■4	Devel	13	08MAR04	24MAR04	10MAR04	24MAR04	08MAR04	22MAR04	.	
□□5	Modul	6	08MAR04	15MAR04	10MAR04	17MAR04	08MAR04	15MAR04	.	
□□6	Modul	5	18MAR04	24MAR04	18MAR04	24MAR04	16MAR04	22MAR04	.	
□7	Docum	8	11MAR04	22MAR04	25MAR04	05APR04	11MAR04	22MAR04	.	
□8	Test	8	25MAR04	05APR04	25MAR04	05APR04	23MAR04	01APR04	.	
□9	Ship	0	06APR04	06APR04	06APR04	06APR04	02APR04	02APR04	.	
□10	UI Me	3	15MAR04	17MAR04	15MAR04	17MAR04	.	.	15MAR04	SGE

If the project delay resulting from the UI Meeting is of concern, you may want to schedule the meeting on an earlier date. Suppose the revised start date of the meeting is March 10, 2004. To see the effect of the change, you can do the following:

1. Revert to the saved project preferences so that both the Table and the Gantt Views are visible.
2. Use the **View** menu to move the Target Date column to the left in the Table View.
3. Scroll down, if necessary, to bring the task ‘UI Meeting’ into view.
4. Change the Target Date column for this task to `'10Mar04.'`

The resulting view is displayed in Output 5.3.4. Note that, as a result of this change, all the activities are back on schedule as the new schedule coincides with the saved baseline schedule. The last activity was defined after the baseline schedule had been saved in Example 5.2; hence, there is no baseline schedule bar for this activity. You can use the **Fill Missing Baseline** selection from the menu shown in Figure 5.26 to set the baseline schedule for the ‘UI Meeting’ to be the current early schedule.

Output 5.3.4 Editing Target Date



Example 5.4: Effect of Calendars

Continuing with the project scenario in the preceding examples, you want to explore other ways of shortening the project duration. One possible alternative is to work overtime. As the project manager, you would like to see the effect on the schedule if you change the calendar for all the development tasks to a six-day calendar.

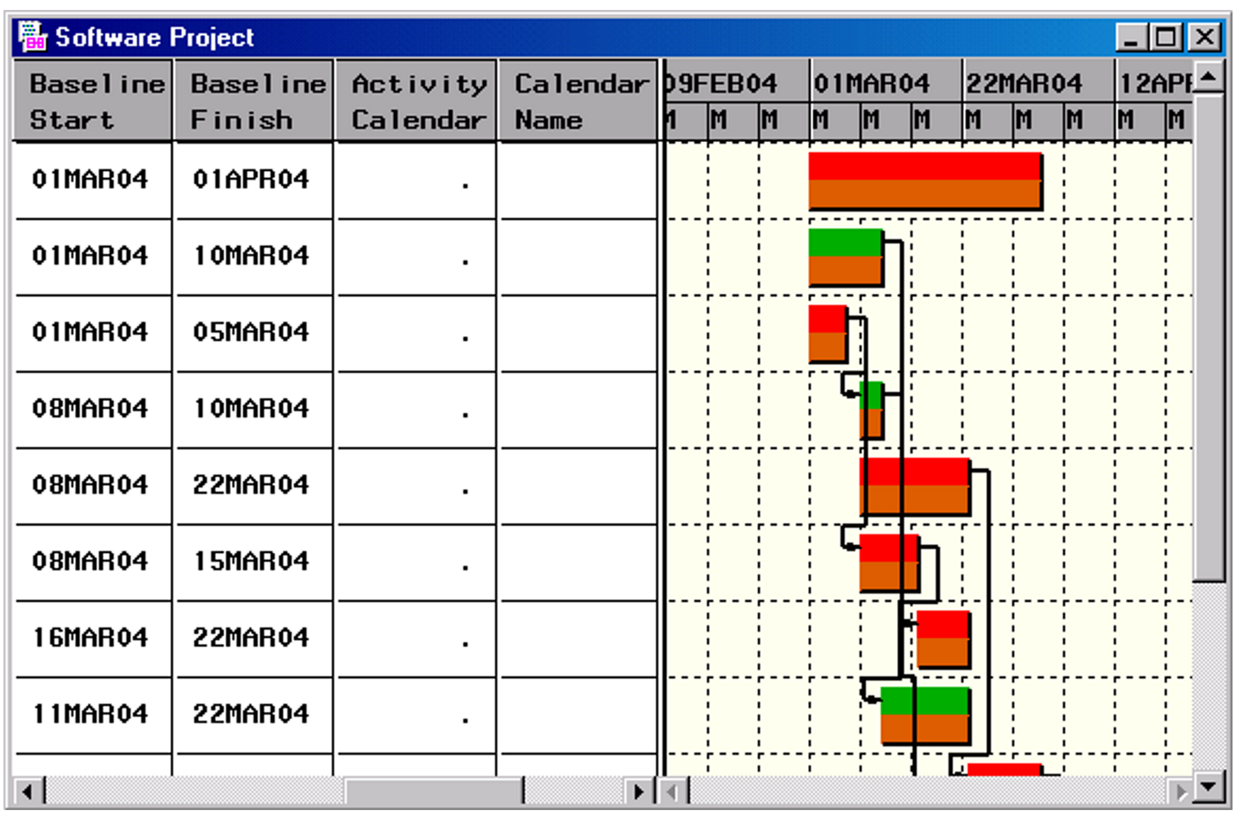
Calendars are defined using the Calendar data set, as in the CPM procedure. This example defines a Calendar data set and invokes PROC PM as follows. Note that, in order to use calendars, the Activity data set needs to have a CALID variable, which is added in a simple DATA step.

```
*  Define a Calendar data set identifying           ;
*  Saturday as a workday                            ;
data calendar;
   input calid calname $ _sat_ $;
   datalines;
1 Sixday WORKDAY
;

*  Add the CALID variable to the Activity data set  ;
*  saved in the preceding example                   ;
data softout3;
   set softout3;
   calid=.;
   run;
```

```
*  Use softout3 as the Activity data set and specify ;
*  the preceding calendar data set                    ;
proc pm data=softout3 project=softattr
        calendar=calendar
        date='1mar04'd interval=weekday
        projname='Software Project'
        out=softout4;
   act actid;
   succ succid;
   project pntid;
   duration duration;
   id activity;
   calid calid;
   run;
```

When the PM procedure initializes the PM window, it attempts to restore all the display settings using the values in the Project data set, SOFTATTR. However, the new Activity data set has an extra variable, calid, which leads to two new columns in the Table View, one for the Activity Calendar (which displays the Calendar ID) and the other for the Calendar Name. These columns are added at the right end of the Table View and can be seen by scrolling to the right. The resulting view is displayed in Output 5.4.1.

Output 5.4.1 Calendar Columns

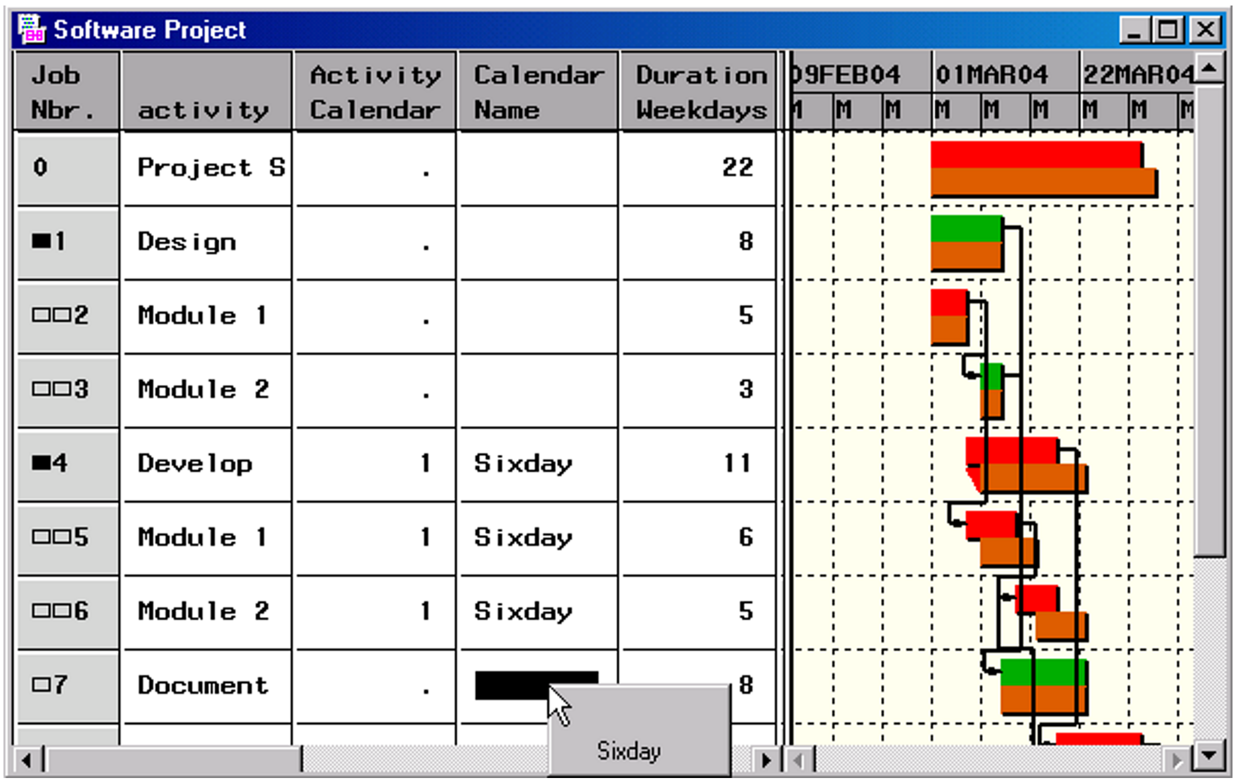


By default, all the activities are assumed to follow the standard five-day calendar. Now, you want to change the calendar for the supertask 'Develop' and all its subtasks to be the six-day calendar defined in the data set CALENDAR. Note that, in the calendar definition, it is sufficient to specify that Saturday is a working day. All the other days of the week default to the default calendar's work pattern; see "Default Calendar" on page 102 in Chapter 4, "The CPM Procedure."

To facilitate the editing of the calendar values and to see the effect on the project duration, reorder the columns (drag the columns in the Table Header) to display the activity, Activity Calendar, Calendar Name, and Duration columns in the Table View. You may need to move the dividing line between the Table and Gantt Views.

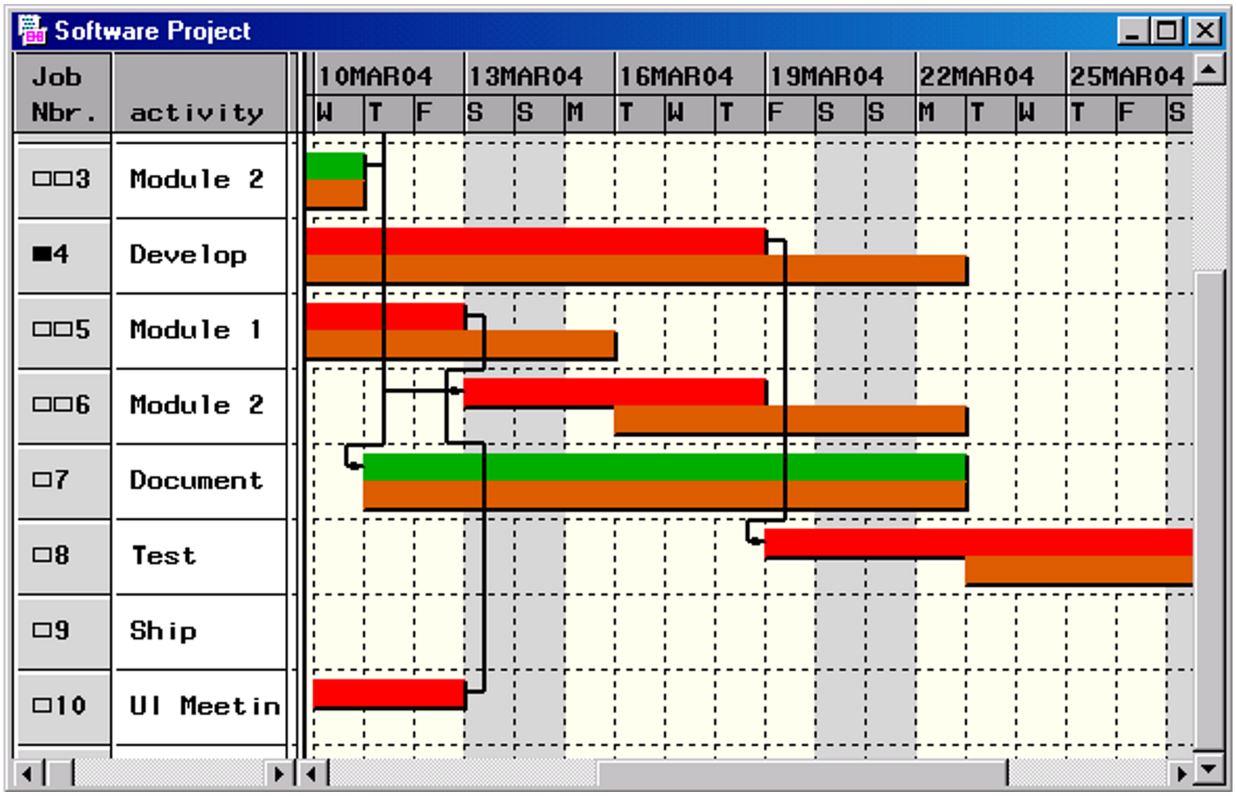
You can enter the Calendar values by typing the number 1 in the Activity Calendar column or the value `'Sixday'` in the Calendar Name column. You can also use the Calendar pop-up menu in one of the calendar columns to select the desired calendar (see Output 5.4.2). Note that the project duration has reduced to 22 days as a result of the six-day calendar.

Output 5.4.2 Effect of Six-Day Calendar



To see the effect on the individual activities, change the units to "Days" in the Gantt View and enlarge the Gantt View, as shown in Output 5.4.3.

Output 5.4.3 Gantt View of Calendar Effect



Example 5.5: Defining Resources

In all the preceding examples, it was assumed that you had enough resources to work on the different tasks. Unfortunately, as a project manager you need to schedule your project using the limited set of resources available to you. In this example, you will assign some project resources and schedule the project subject to resource constraints.

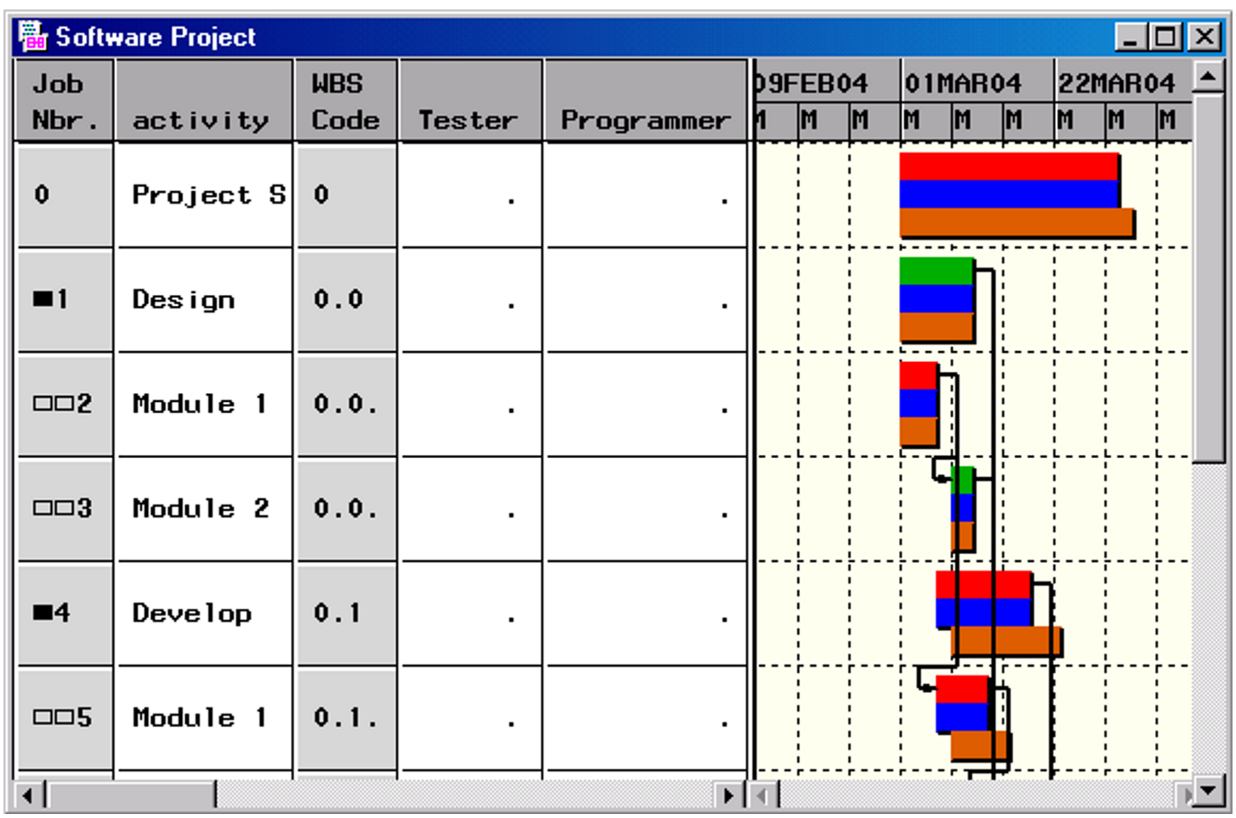
In order to assign resources to the tasks, you need to add resource variables to the Activity data set as well as define a resource availability (Resource) data set.

Suppose that the resources that you are interested in are Tester and Programmer. The following statements set up the project data needed to perform resource-constrained scheduling with PROC PM using the output data produced in Example 5.4.

```
*  Define a Resource data set specifying   ;
*  1 Tester and 1 Programmer as the        ;
*  available resources                     ;
data resources;
   input _date_ date7. Tester  Programmer;
   datalines;
01jan04   1   1
;

*  Add the resource variables Tester and       ;
*  Programmer to the Activity data set         ;
```

```
*  (the output data set saved in last example) ;
data softout4;
   set softout4;
   Tester=.;
   Programmer=.;
   run;

*  Use softout4 as the Activity data set and    ;
*  specify the preceding Resource data set.     ;
*  Save the schedule in softout5.               ;
proc pm data=softout4 project=softattr
        calendar=calendar
        resourcein=resources
        date='1mar04'd interval=weekday
        projname='Software Project'
        out=softout5;
   act actid;
   succ succid;
   project pntid;
   duration duration;
   id activity;
   calid calid;
   resource Tester Programmer / per=_date_;
   run;
```

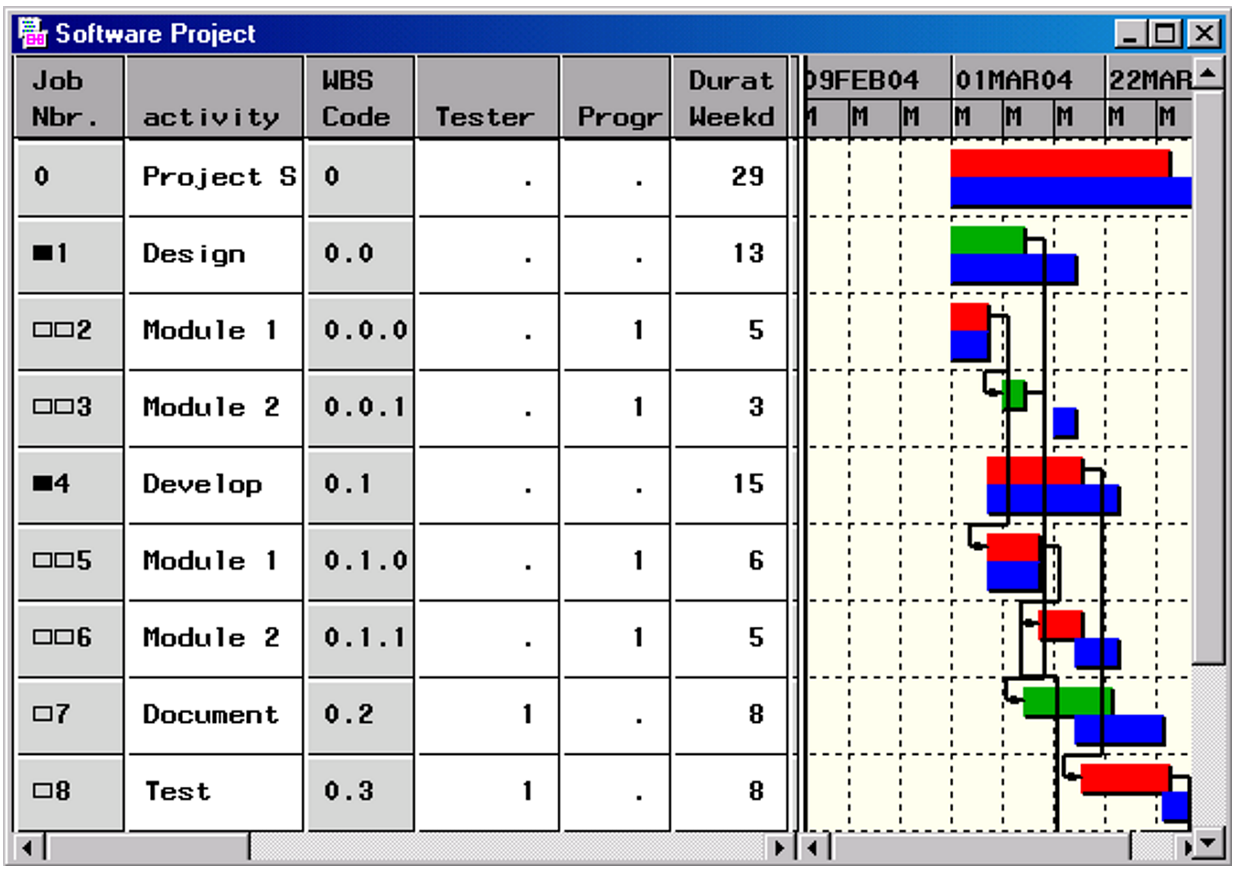
Output 5.5.1 Adding Resources to the Project



Output 5.5.1 shows the Table and Gantt Views of the project after rearranging some of the columns and moving the dividing line to show the resource columns. The Resource Schedule bars are also brought into display by right-clicking on the background in the Gantt View and selecting **Resource Schedule**. The Resource Schedule bar is shown (in blue) between the Early Schedule bar and the Baseline Schedule bar. Note that the resource schedule coincides with the early schedule because no resource requirements have been specified for either of the two resources.

You can now use the Table View to enter the resource requirements for each task. Set the requirement for the resource Tester to `'1'` for the tasks 'Document' and 'Test,' and the requirement for the resource Programmer to '1' for the tasks numbered '2,' '3,' '5,' and '6.' The resulting schedule is displayed in Output 5.5.2. In this view, the baseline schedule is not displayed. You can see that several of the tasks have been delayed, resulting in lengthening the project duration to 29 weekdays.

Output 5.5.2 Editing Resource Requirements



You can set the resource-constrained schedule as a baseline to do some “what-if” analysis. For example, suppose you would like to determine the effect of adding another programmer to the project. In order to change the resource availability, you need to save the current project, edit the Resource availability data set to add another programmer, and then reinvoke the PM procedure.

First, in the PM window displayed in Output 5.5.2, do the following:

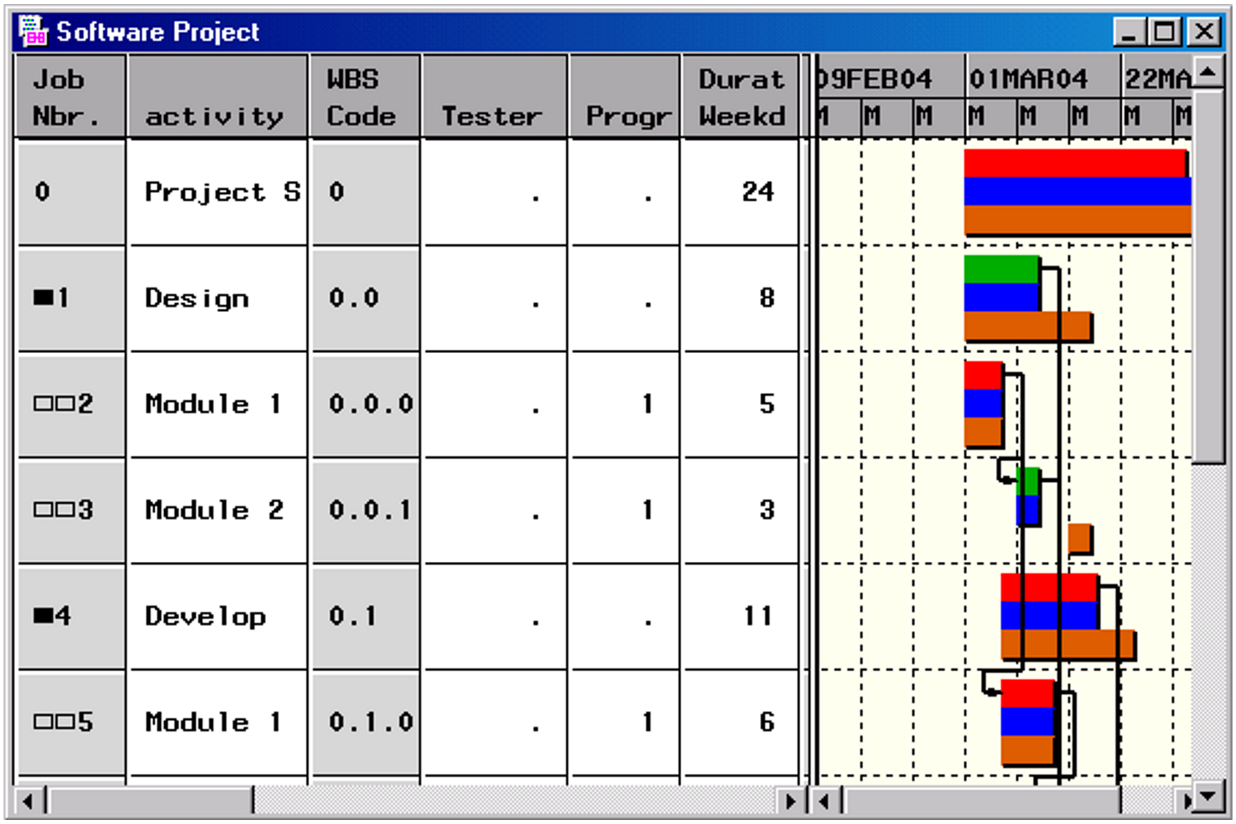
1. Display the Baseline Schedule bar again.
2. Use the **Replace Baseline** selection from the **Edit** menu to select the Resource Schedule as the new baseline schedule.
3. Save the project preferences.
4. Close the PM window.

Use the following statements to reinvoke PROC PM after defining a new resource availability:

```
*  Change the resource availability for Programmer to 2 ;
data resources;
   input _date_ date7. Tester  Programmer;
   datalines;
01jan04  1  2
;

*  Use softout4 as the Activity data set and specify ;
*  the preceding Resource data set.                  ;
*  Save the schedule in softout5.                    ;
proc pm data=softout4 project=softattr
        calendar=calendar
        resourcein=resources
        date='1mar04'd interval=weekday
        projname='Software Project'
        out=softout5;
   act actid;
   succ succid;
   project pntid;
   duration duration;
   id activity;
   calid calid;
   resource Tester Programmer / per=_date_;
   run;
```

Using the new resource availability, you reduced the project duration by five days. The resulting schedule is displayed in Output 5.5.3, which also shows the baseline schedule corresponding to the earlier resource availability data set.

Output 5.5.3 Comparison of Two Resource Schedules



Example 5.6: Editing Progress

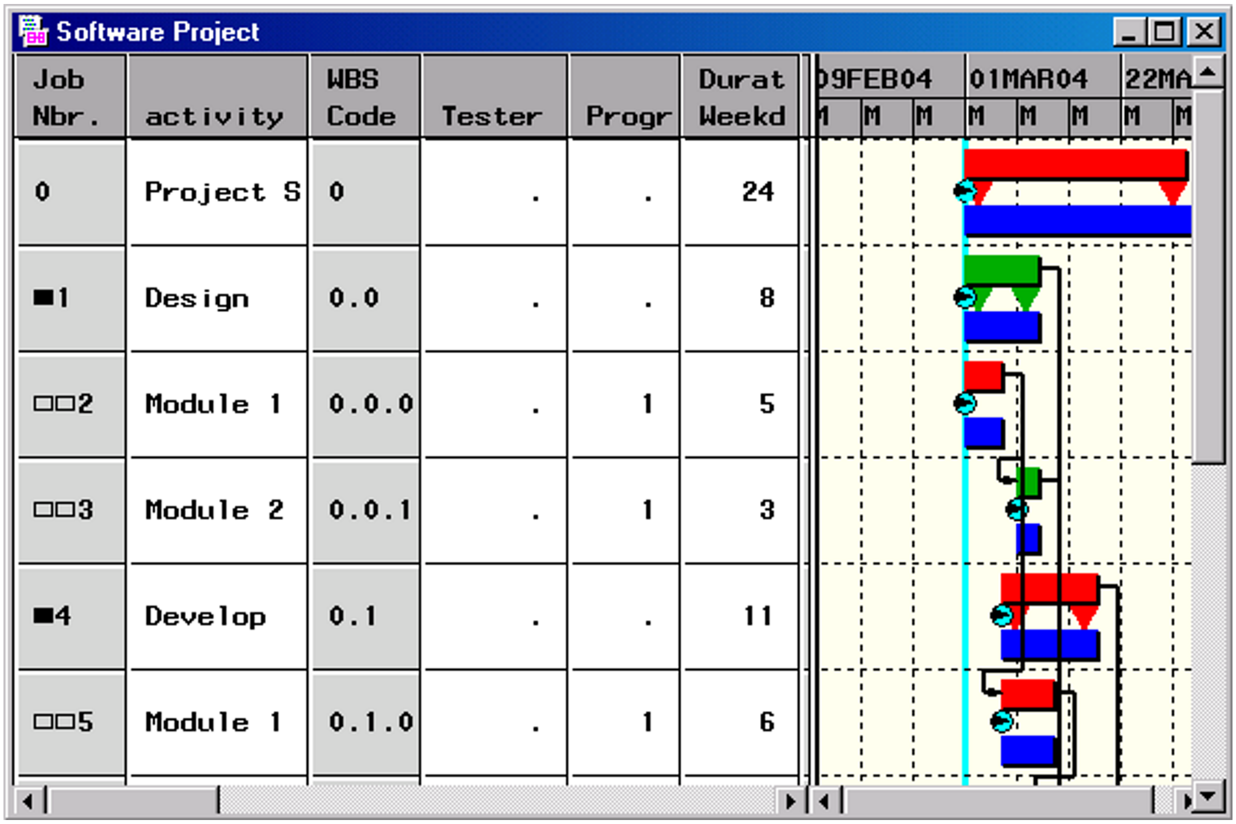
Once a project plan has been established and the project is under way, a major part of a project manager's responsibility is to monitor the project as it progresses. This example uses the PM window to add progress information to the project and discusses some of the related editing functions.

In the final window of Example 5.5 (shown in Output 5.5.3), do the following:

1. Delete the baseline schedule using the **Edit** menu.
2. From the **Edit** menu, select **Add Progress**.

The resulting display is shown in Output 5.6.1. The Gantt View now shows the Actual Schedule bar between the Early Schedule bar and the Resource Schedule bar. It also displays a Timenow Line. Since no progress information has been entered, the Timenow Line is drawn at the beginning of the project and all the Actual Schedule bars show only a handle that can be used to drag progress for a particular task.

Output 5.6.1 Adding Progress Information to Project

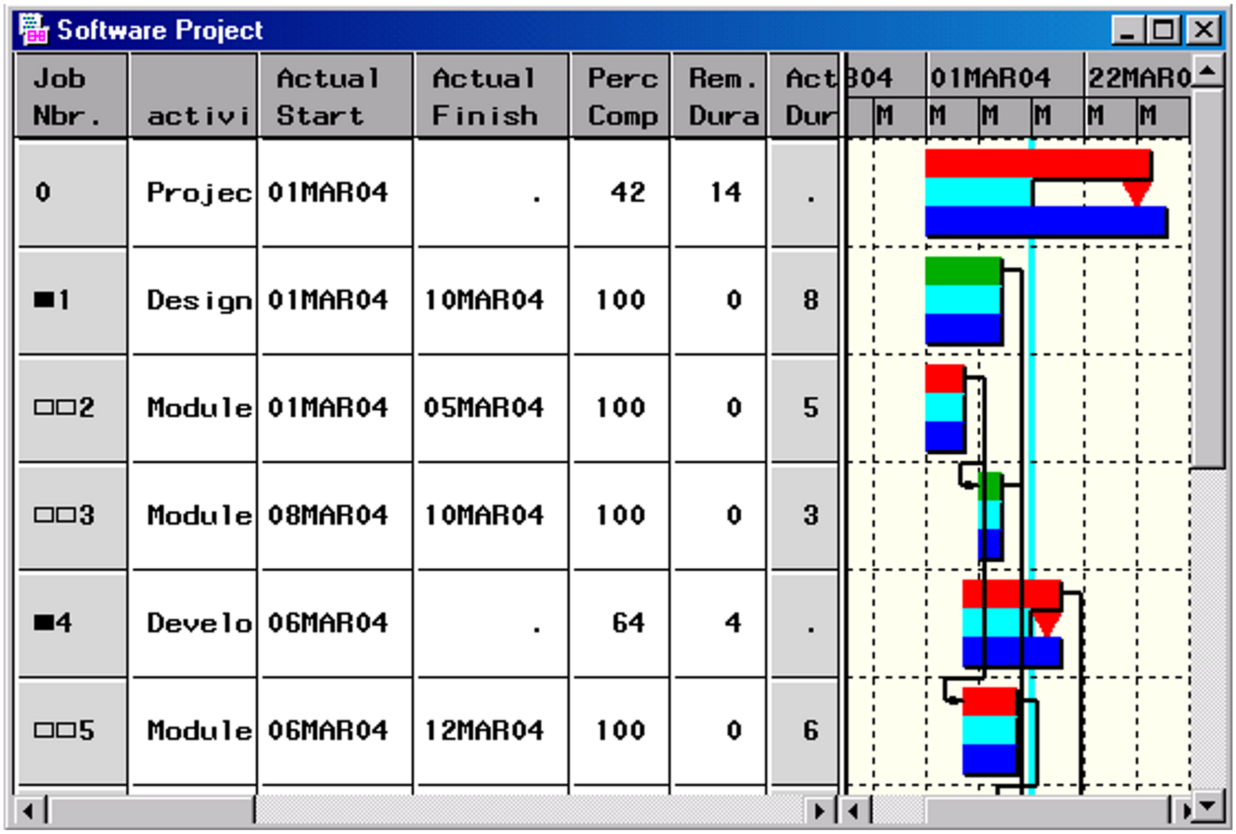


You can enter progress information in several ways:

- Drag the Timenow Line to update progress information for several tasks at once. The actual start and finish times (until the Timenow date) are set assuming that the tasks follow the resource-constrained schedule. (If there are no resource constraints, the tasks are assumed to follow the early schedule.)
- Use the handle on the Actual Schedule bar for a given task to drag the progress bar.
- Bring the Progress columns into view in the Table View and edit one of the Progress columns.

As an example, drag the Timenow Line to the tick mark corresponding to 15MAR04. The resulting window (after reordering and resizing some columns and scrolling the Gantt View) is shown in Output 5.6.2.

Output 5.6.2 Moving the Timenow Line



Note that some of the activities are completed while others are still in progress. If the project data are saved at this point, the Schedule data set will have all the Progress variables (A_START, A_FINISH, PCT_COMP, and REM_DUR). However, for the PM procedure to be able to recapture the exact state of the schedule as it was saved, it also needs to know the value of TIMENOW when the project data was last saved. This value (`'15Mar04'` for the current example) is saved as a macro variable named TIMENOW (see the section "Macro Variable TIMENOW" on page 328).

To see how the Actual information can be used from one invocation of PROC PM to the other, save the project as displayed in Output 5.6.2 and then reinvoke PROC PM to continue editing the progress information.

Recall from the last invocation of PROC PM that the data are saved in the data set SOFTOUT5. To use the saved progress information, invoke PROC PM as follows:

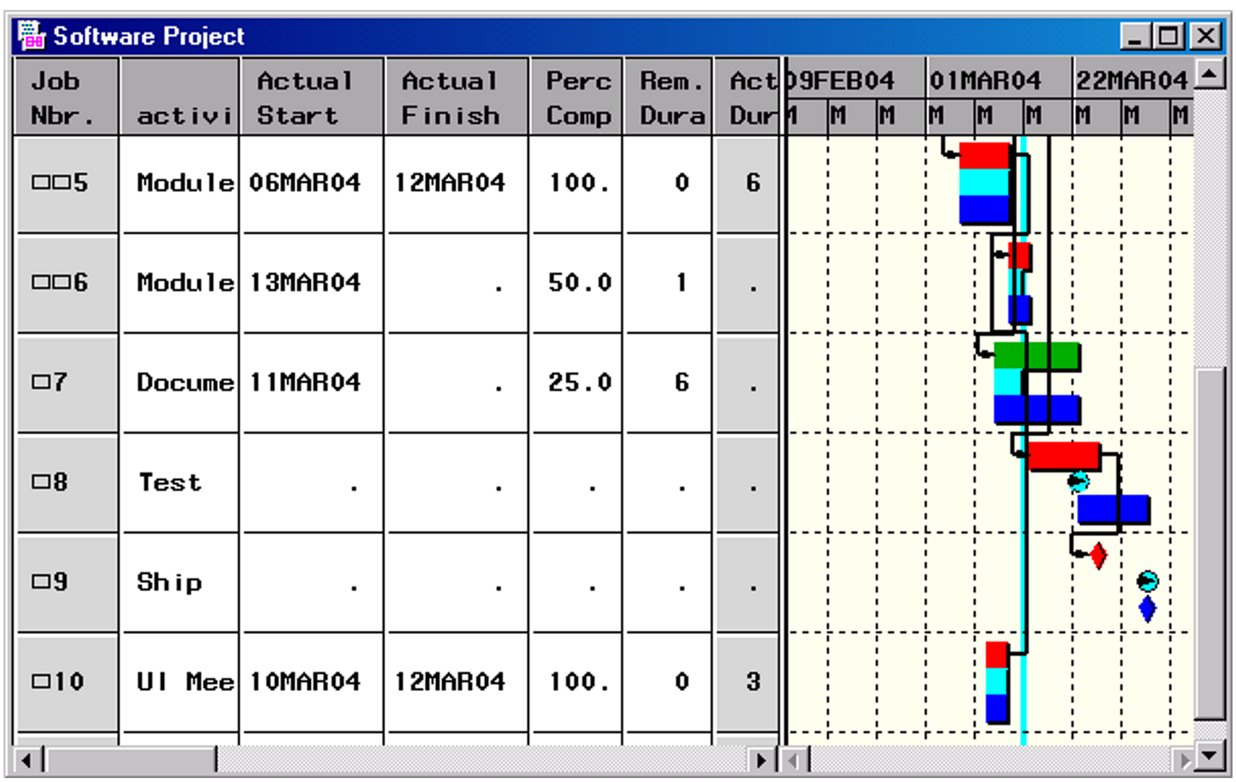
```
* Use softout5 as the Activity data set and specify  ;
* the Resource data set defined in the last example. ;
* Save the schedule in softout6.                     ;
proc pm data=softout5 project=softattr
        calendar=calendar
        resourcein=resources
        date='1mar04'd interval=weekday
        projname='Software Project'
        out=softout6;
   act actid;
   succ succid;
   project pntid;
```

```
duration duration;
id activity;
calid calid;
*  Use the ACTUAL statement to specify the Progress variables  ;
*  and the value of TIMENOW saved from the previous invocation ;
actual / as=a_start af=a_finish
         remdur=rem_dur pctcomp=pct_comp
         timenow=&timenow;
resource Tester Programmer / per=_date_;
run;
```

The preceding program displays the PM window for the updated Software project. Now use the Table View to edit some of the Progress columns. To do so, you can either scroll to the Progress Columns or move these columns to the left in the Table View using the appropriate selection from the **View** menu (Figure 5.9).

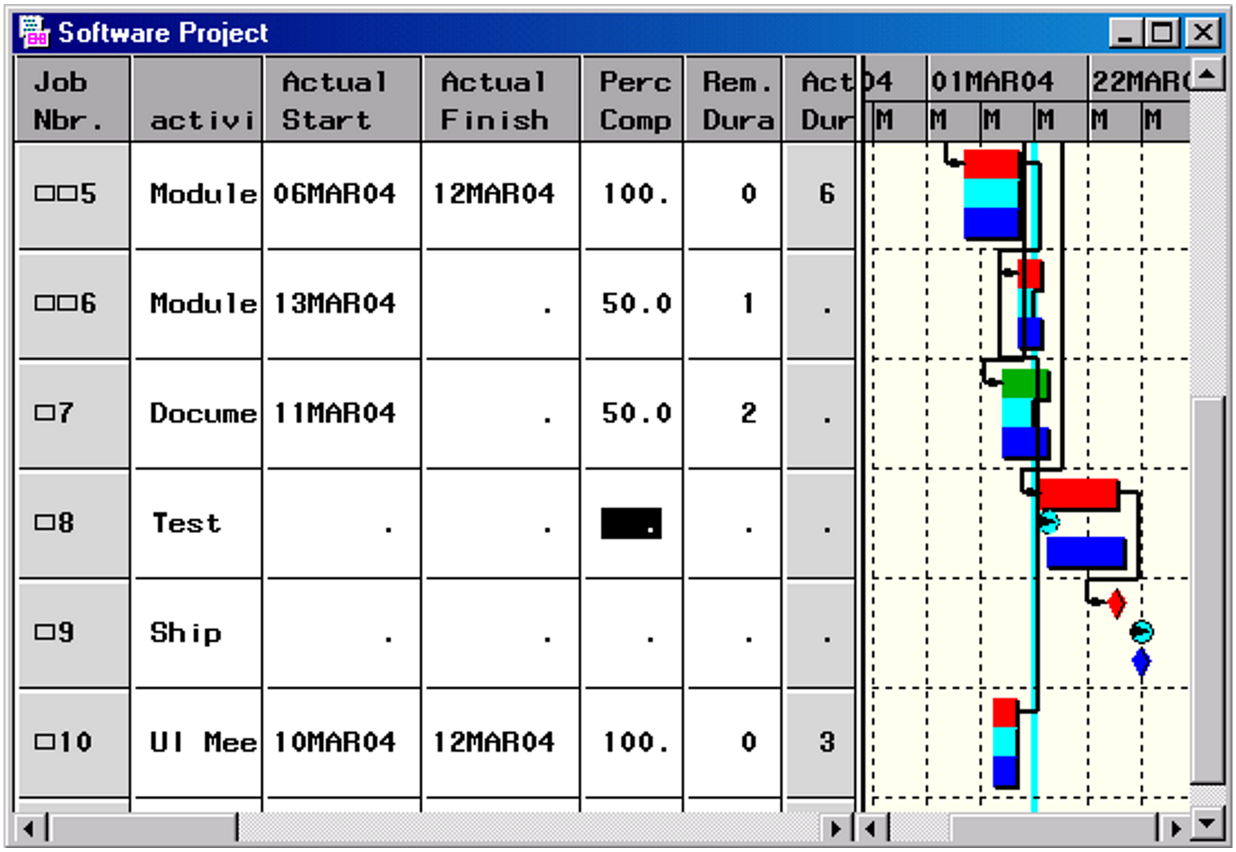
Task number 6 ('Module 2' under 'Develop') has a Remaining Duration value of 4. Now, you may notice that you have misjudged the amount of work involved and that you need only one more day to finish the task. Enter 1 in the Remaining Duration column. This editing change immediately causes the Percent Complete column to update to 50, indicating that 50% of the work is completed. The resulting effect on the project schedule is shown in Output 5.6.3 (the window has been scrolled down so that the second half of the project is visible). Note that reducing the duration of the 'Module' task did not affect the project end date; the duration of the project is still 24 days. Studying the schedule of the 'Document' and 'Test' tasks, you notice that the delay to the project is caused by the fact that the resource-constrained schedule of the task 'Test' is delayed due to resource constraints.

Output 5.6.3 Editing the Remaining Duration Column



In addition to revising the progress information for 'Module 2,' you also realize that the 'Document' task is 50% complete as of the Timenow date. Edit the Percent Complete column in the Table View, changing the value from 25.0 to 50.0. Immediately, the Remaining Duration column changes to 2. The resulting window is shown in Output 5.6.4. The project end date (for the resource-constrained schedule) is 28Mar04. Thus, the project duration is now reduced to 20 days.

Output 5.6.4 Editing the Percent Complete Column



Chapter 6
The Microsoft Project Conversion Macros

Contents

Overview: Microsoft Project Conversion Macros

The SAS macro %MSPTOSAS is available for converting Microsoft Project data to a form that is readable by the PM procedure. The %MSPTOSAS macro converts Microsoft Project 98, 2000, 2002, 2003, 2007, and 2010 data. The macro generates the necessary SAS data sets, determines the values of the relevant options, and enables you to invoke an instance of the PM procedure with the converted project data. Execution of this macro requires SAS/ACCESS Interface to PC File Formats software.

The %SASTOMSP macro converts data sets used by the CPM and PM procedures into a form that is readable by Microsoft Project 2000, 2002, and 2003.

NOTE: Microsoft Project 98, 2000, 2002, 2003, 2007, and 2010 are registered trademarks of Microsoft Corporation.

%MSPTOSAS

%MSPTOSAS is a SAS macro that converts Microsoft Project data saved in an MDB (Microsoft Access database) or XML format into data sets that are readable by the PM procedure. %MSPTOSAS requires specification of the location and name of either the MDB or the XML file (but not both), and it requires specification of the version of Microsoft Project that was used to create input. You can also include optional arguments to specify a location for storing SAS data sets, to control the mode of the PM procedure invocation, or to run the %MSPTOSAS macro on UNIX.

The %MSPTOSAS macro converts the hierarchical relationships, precedence relationships, time constraints, resource availabilities, resource requirements, project calendars, task calendars, resource calendars, holiday information, work-shift information, actual start and finish times, and baseline start and finish times. In addition, the task custom fields are extracted and stored in the Task_Attributes data set.

%MSPTOSAS Macro Parameters

%MSPTOSAS (**LIBRARY=***library*, **MAPFILE=***mapfile*,
MDBFILE=*mdbfile*, **VERSION=***version*,
VIEWPM=*indicator*, **XMLFILE=***xmlfile*,
DBMS=*identifier*, **SERVER=***pc-server-hostname*,
PORT=*port-number*);

General Project Parameters

LIBRARY=*library*

specifies the location of the directory for storing the SAS data sets and the file *callpm.sas*. This parameter can be either a SAS library reference or a path. A path can be specified with double quotes, single quotes, or no quotes. See the MDBFILE= parameter for an example. After a successful run of the %MSPTOSAS macro, the library reference ‘mspout’ is assigned to this path. The default value of the LIBRARY= parameter is WORK, which is the default libref that is generally assigned to your temporary SAS data library for the current session or job.

MDBFILE=*mdbfile*

specifies the location and filename of the MDB file. This parameter can be either a SAS file reference or a filename. A filename can be specified with double quotes, single quotes, or no quotes. One of MDBFILE= or XMLFILE= is required, but they cannot both be specified.

XMLFILE=*xmlfile*

specifies the location and filename of the XML file. This parameter can be either a SAS file reference or a filename. A filename can be specified with double quotes, single quotes, or no quotes. One of MDBFILE= or XMLFILE= is required, but they cannot both be specified.

MAPFILE=*mapfile*

specifies the location and filename of the XML map file. The map file is a separate XML document that contains specific XML map syntax. The map file tells the XML engine how to interpret the XML markup in order to successfully import the XML document. This parameter can only be used when XMLFILE= is specified. The value of MAPFILE= can be either a SAS file reference or a filename. A filename can be specified with double quotes, single quotes, or no quotes. If XMLFILE= is specified and MAPFILE= is not, a map file named mspxml.map will be created in the WORK directory and used as the XML map file. For more information related to the XML engine and XML map syntax, refer to SAS XML LIBNAME Engine User's Guide.

VERSION=*version*
specifies the version of Microsoft Project used to create the MDB or XML file. The supported versions for MDB files are 98, 2000, 2002 and 2003. The supported versions for XML files are 2002, 2003, 2007, and 2010. This parameter is required; there is no default value. For example, to convert an MDB file from Microsoft Project 2003 format, you need to specify VERSION=2003.

VIEWPM=*indicator*
controls the mode of the PM procedure invocation. The value VIEWPM=1 invokes the PM procedure in interactive mode, and the value VIEWPM=0 invokes the PM procedure in NODISPLAY mode. This parameter is optional; the default value is 1.

UNIX Parameters

The following statements are available to establish a connection from SAS running on UNIX to a SAS PC Files Server. This enables you to run the %MSPTOSAS macro on UNIX. For more information, see *SAS/ACCESS Interface to PC Files: Reference*.

DBMS=*identifier*
specifies the type of data to import. In this case, the value DBMS=accesscs must be specified.

PORT=*port-number*
specifies the port or service name that the SAS PC Files Server is listening to on the PC. The default value is 8621.

SERVER=*pc-server-hostname*
specifies the name of the computer on which you started the PC files server. This name is required by UNIX users to connect to this server machine and is reflected on the server control panel. This hostname can be specified as a simple computer name (e.g., wxp320), a fully qualified network name (e.g., `wxp320.domain.com`), or an IP address.

%MSPTOSAS Macro Output

Once the file conversion is complete, the %MSPTOSAS macro uses the information from the MDB or XML file in a call to the PM procedure. The %MSPTOSAS conversion macro generates the following SAS data sets:

- Activity data set: Activity
- Calendar data set: Calendar
- Holiday data set: Holiday
- Workday data set: Workday

- Resource data set: Resource
- Schedule data set: Schedule
- Task attributes data set: Task_Attributes
- Preferences data set: Prefs

In addition, the macro generates the SAS code that enables you to reinstate your project in current and future SAS sessions. The name of the file containing this SAS code is callpm.sas. The data sets and callpm.sas file are created in the location specified by the LIBRARY= parameter. For more information about data sets, variables, options, and statements related to the PM procedure, refer to Chapter 5, "The PM Procedure."

The following statements are taken from a callpm.sas file:

```
Libname mspout "C:\MSPROJ";
PROC PM data = mspout.Activity project=mspout.Prefs
     caledata = mspout.Calendar
     workdata = mspout.Workday
     out=mspout.Schedule
     interval=dtday
     date="17DEC06:08:00:00"dt
     daylength=" 8:00"t
     suppressobswarn
     setfinishmilestone;
activity ACTID;
successor SUCCUID / LAG = LAG;
duration DURATION;
project PNTUID;
id ACTIVITY ACTUID;
run;
```

Activity Data Set

Table 6.1 lists and describes the variables in the data set Activity. For more information about the activity data set in the PM procedure, see the section "Input Data Sets and Related Variables" on page 135 in Chapter 4, "The CPM Procedure."

Table 6.1 Variables in the Data Set Activity

Variable	Type	Description
CAL	Numeric	Activity calendar variable
A_START	Numeric	Actual start variable
A_FINISH	Numeric	Actual finish variable
ACTID	Numeric	Activity variable; has different values in the data set schedule
ACTIVITY	Character	Activity name variable; used as an ID variable in the PM procedure
ACTUID	Numeric	Variable uniquely identifying activities in the data sets Activity, Schedule, and Task_Attributes
ALGNDATE	Numeric	Alignment date variable

Table 6.1 (continued)

Variable	Type	Description
ALGNTYPE	Character	Alignment type variable
ASSN_WORK	Numeric	Resource work variable
B_START	Numeric	Baseline start date variable
B_FINISH	Numeric	Baseline finish date variable
DURATION	Numeric	Duration variable
LAG	Character	Lag variable
PNTUID	Numeric	Parent variable
SUCCUID	Numeric	Successor variable
TASK_ID	Numeric	Position identifier of the current activity in the list of activities

Calendar Data Set

The data set Calendar contains basic calendar data used by the PM and CPM procedures. If a task has "elapsed" duration in Microsoft Project, a special calendar is defined on 24-hour days and 7-day weeks, including holidays and other nonworking days. This calendar is associated with the tasks that have "elapsed" durations. For more information, see the section "Multiple Calendars" on page 101 in Chapter 4, "The CPM Procedure."

Holiday Data Set

Table 6.2 lists and describes the variables in the data set Holiday. For more information, see the section "Multiple Calendars" on page 101 in Chapter 4, "The CPM Procedure."

Table 6.2 Variables in the Data Set Holiday

Variable	Type	Description
CAL	Numeric	Holiday calendar variable
HFINISH	Numeric	Holiday finish variable
HOLIDUR	Numeric	Holiday duration variable
HSTART	Numeric	Holiday start variable

Workday Data Set

Each variable in the data set WORKDAY defines a shift in the workday. For more information, see the section "Multiple Calendars" on page 101 in Chapter 4, "The CPM Procedure."

Resource Availability Data Set

The data set Resource contains information about the resources required by the activities in the project. In addition to resource variables, the variables in Table 6.3 are included in the data set Resource. For more information, see the section "Resource Usage and Allocation" on page 113 in Chapter 4, "The CPM Procedure."

Table 6.3 Variables in the Data Set Resource

Variable	Type	Description
AVAILDTE	Numeric	Resource availability date/time variable
OBSTYPE	Character	Observation type variable

Schedule Data Set

The data set Schedule contains the schedule determined by the PM procedure. This data set is described in the section "Schedule Data Set" on page 329 in Chapter 5, "The PM Procedure." In addition to the standard schedule variables, the Schedule data set created by the %MSPTOSAS macro contains the variables ACTUID and ACTIVITY. Since ACTUID and ACTIVITY are also present in the Activity and Task_Attributes data sets, these variables enable you to identify activities and select a subset of the variables in these three data sets to create your report.

Task Attributes Data Set

The data set Task_Attributes stores task attribute variables. The variables in Table 6.4 take their names from Microsoft Project custom fields. If you have renamed a custom field in Microsoft Project, then the name you specified for that field is used as the label for the corresponding SAS variable. A custom field can be converted only if it has at least nondefault values for at least one task or if the field has been renamed in Microsoft Project. The variables in the data set Task_Attributes can have enterprise versions (e.g., Enterprise Cost1) and enterprise project versions (e.g., Enterprise Proj Cost1) if your project has enterprise fields defined with the Microsoft Enterprise Global Template and Enterprise Project Versions.

Table 6.4 Variables in the Data Set Task_Attributes

Variable	Type	Description
Cost1–10	Numeric	Custom cost fields in Microsoft Project
Date1–30	Numeric	Custom date fields in Microsoft Project
Duration1–10	Numeric	Custom duration fields in Microsoft Project
Finish1–10	Numeric	Custom finish date/time fields in Microsoft Project
Flag1–20	Character	Custom flag fields in Microsoft Project
Number1–20	Numeric	Custom number fields in Microsoft Project
Outlinecode1–10	Character	Custom outline code fields in Microsoft Project
Start1–10	Numeric	Custom start date/time fields in Microsoft Project
Text1–10	Character	Custom text fields in Microsoft Project

In addition to the variables in the preceding table, the data set Task_Attributes converts the following variables from Microsoft Project: HYPERLINK, HYPERLINKADDRESS, HYPERLINKSUBADDRESS, NOTES, TASK_COST, TASK_WBS, TASK_PCT_COMP, TASK_PCT_WORK_COMP, TASK_PRIORITY, and TASK_REM_DUR.

In the data set Task_Attributes, the variable DURATION_ELAPSED indicates whether the task duration is "elapsed" in Microsoft Project. When a task duration is elapsed, the special calendar *n*099999 is associated with this task in the Activity data set. See the section "Calendar Data Set" on page 359 for details.

Finally, the data set Task_Attributes contains the ACTUID and ACTIVITY variables. These variables are also contained in the data sets Activity and Schedule.

NOTE: For MDB files created in Microsoft Project 98, the only variables present in the data set Task_Attributes are ACTUID, ACTIVITY, DURATION_ELAPSED, TASK _COST, TASK_PCT_COMP, TASK_PCT_WORK_COMP, TASK_PRIORITY, TASK _REM_DUR, and TASK_WBS.

Project Preference Data Set

The data set Prefs is created by the PM procedure to save and restore preferences that control the project view. For more information, see the section "Saving and Restoring Preferences" on page 325 in Chapter 5, "The PM Procedure."

%SASTOMSP

%SASTOMSP is a SAS macro that converts data sets used by the PM and CPM procedures into a file that is readable by Microsoft Project 2000, 2002, and 2003.

NOTE: The MDB file created by the %SASTOMSP macro cannot be read directly by versions of Microsoft Project dated 2007 or later. This issue can be circumvented by opening the MDB file in Microsoft Project 2003, then saving the file in a format supported by versions of Microsoft Project dated 2007 or later (.mpp or .xml, for example).

The macro converts information that is common to both the CPM and PM procedures and Microsoft Project, including hierarchical relationships, precedence relationships, time constraints, resource availabilities, resource requirements, project calendars, resource calendars, task calendars, holiday information, and workshift information. In addition, the early and late schedules, the actual schedule, the resource-constrained schedule, and the baseline schedule are also extracted and stored as start-finish fields.

%SASTOMSP Macro Parameters

%SASTOMSP(**PROJ_NAME=***project_name*, **LIBRARY=***library*,
MDBFILE=*mdbfile*, **ACTDS=***SAS-data-set*,
CALDS=*SAS-data-set*, **HOLDS=***SAS-data-set*,
WORKDS=*SAS-data-set*, **RESDS=***SAS-data-set*,
SCHEDULEDS=*SAS-data-set*, **_INTERVAL=***_interval*,
_DATE=*_date*, **_FBDATE=***_fbdate*,
_DAYSTART=*_daystart*, **_DAYLENGTH=***_daylength*,
_CALID=*variable*, **_ACTIVITY=***variable*,
_ID=*variable*, **_DUR=***variable*,
_ALIGNDATE=*variable*, **_ALIGNTYPE=***variable*,
_HEAD=*variable*, **_TAIL=***variable*, **_PROJECT=***variable*,
_SUCCESSOR=*_successor*, **_LAG=***_lag*,
_HOLISTART=*variable*, **_HOLIEND=***variable*,
_HOLIDUR=*variable*, **_RESOURCE=***_resource*,
_RESOBSTYPE=*variable*, **_RESPERIOD=***variable*,
DBMS=*identifier*, **SERVER=***pc-server-hostname*,
PORT=*port-number*);

General Project Parameters

LIBRARY=*library*

specifies the location of the input SAS data sets. This parameter can be either a SAS library reference or a path. A path can be specified with double quotes, single quotes, or no quotes. For example, in the following %SASTOMSP invocation, a path is specified with double quotes:

```
%sastomsp(library="C:\SASPROJ", mdbfile=C:\MSPROJ\filename.mdb);
```

The default value is 'WORK', which is the default libref generally assigned to your temporary SAS data library for the current session or job.

MDBFILE=*mdbfile*

specifies the output MDB file. This parameter can be either a SAS fileref or a filename. A filename can be specified with double quotes, single quotes, or no quotes.

If *mdbfile* does not have the file extension .mdb, the macro checks whether it is an assigned fileref. If *mdbfile* is a fileref, the macro creates an MDB file as specified by this fileref. If *mdbfile* is not a fileref, the resulting MDB file takes the value of *mdbfile* as the prefix and is created in the location specified by the LIBRARY= parameter.

If *mdbfile* is a fileref, the macro creates an MDB file as specified by this fileref. If you are running the %MSPTOSAS macro on UNIX, *mdbfile* has to be a filename.

If *mdbfile* is a filename without a path, then the %SASTOMSP macro uses the default path of the OUTFILE= option of PROC EXPORT. In many cases, this default path is 'C:\Documents and Settings\username\'. See the PROC EXPORT documentation in *Base SAS Procedures Guide* for details.

This parameter is required; there is no default value.

PROJ_NAME=*project_name*

specifies the name of the project. The default value is 'Project Imported from SAS'.

Input Data Set Parameters

ACTDS=*SAS-data-set*

specifies the name of the Activity data set used by the PM and CPM procedures. The default value is ACTIVITY. For more information, see the section "Input Data Sets and Related Variables" on page 135 in Chapter 4, "The CPM Procedure."

CALDS=*SAS-data-set*

specifies the name of the Calendar data set used by the PM and CPM procedures. There is no default value. This parameter is required when the project uses special calendars. If the project uses the standard calendar defined by the INTERVAL=, DAYSTART=, and DAYLENGTH= parameters, then the CALDS= parameter is optional. For more information, see the section "Multiple Calendars" on page 101 in Chapter 4, "The CPM Procedure."

HOLDS=*SAS-data-set*

specifies the name of the Holiday data set used by the PM and CPM procedures. There is no default value. This parameter is required only if the project uses holidays defined in a Holiday data set. For more information, see the section "Multiple Calendars" on page 101 in Chapter 4, "The CPM Procedure."

RESDS=*SAS-data-set*

specifies the name of the Resource data set used by the PM and CPM procedures. There is no default value. This parameter is required only if the project requires resource information defined in a Resource data set. For more information, see the section "Resource Usage and Allocation" on page 113 in Chapter 4, "The CPM Procedure."

SCHEDULEDS=*SAS-data-set*

specifies the name of the Schedule data set generated by the PM procedure, as described in Chapter 5. You can alternatively specify a Schedule data set generated by the CPM procedure, as described in the section "Schedule Data Set" on page 329 in Chapter 5, "The PM Procedure." The SCHEDULEDS= parameter has no default value. When this parameter is specified, the macro stores the SAS schedules in Microsoft Project date fields, as shown in Table 6.5.

Table 6.5 SAS Schedule Conversion

Schedule	SAS Variables	Microsoft Project Fields
Early schedule	E_START, E_FINISH	SAS_E_start, SAS_E_finish
Late schedule	L_START, L_FINISH	SAS_L_start, SAS_L_finish
Resource-constrained schedule	S_START, S_FINISH	SAS_R_start, SAS_R_finish
Actual schedule	A_START, A_FINISH	SAS_A_start, SAS_A_finish
Baseline schedule	B_START, B_FINISH	SAS_B_start, SAS_B_finish

In Microsoft Project, you can display these date fields by going to the **Insert** menu, selecting **Column**, and selecting the fields you want to display. You can also display the schedule in a Gantt chart. See Example 6.8 for details.

WORKDS=*SAS-data-set*

specifies the name of the Workday data set used by the PM and CPM procedures. There is no default value. This parameter is required only when the specified calendar refers to special work shifts defined in a Workday data set. For more information, see the section "Multiple Calendars" on page 101 in Chapter 4, "The CPM Procedure."

Variable and Option Parameters

_ACTIVITY=*variable*

specifies the name of the variable in the Activity data set that contains the names of the tasks. The default value is ACTIVITY.

_ALIGNDATE=*variable*

identifies the variable in the Activity data set that specifies the date to be used to constrain an activity to start or finish on a particular date. This variable is optional, and it has no default value.

_ALIGNTYPE=*variable*

identifies the variable in the Activity data set that specifies whether the date value specified in the ALIGNDATE= parameter is the earliest start date, the latest finish date, and so forth, for the activity in the observation. This variable is optional, and it has no default value.

_CALID=*variable*
identifies the variable that is used in the Activity, Holiday, and Calendar data sets to identify the calendar to which each observation refers. When this parameter is not specified, the macro uses a default variable named _CAL_ in each of the three data sets.

_DATE=*_date*
specifies the SAS date, time, or datetime that is to be used as an alignment date for the project. The default value is the current value of the SAS system time.

_DAYLENGTH=*_daylength*
specifies the length of the workday. The default value is `'8:00't`, if _INTERVAL is specified as ‘WORKDAY’ or ‘DTWRKDAY’. Otherwise, the default value is `'24:00't`.

_DAYSTART=*_daystart*
specifies the start of the workday. The default value is `'9:00't`, if _INTERVAL is specified as ‘WORKDAY’ or ‘DTWRKDAY’. Otherwise, the default value is `'0:00't`.

_DUR=*variable*
identifies the variable in the Activity data set that contains the durations of the tasks. The default value is ‘DUR’.

_FBDATE=*_fbdate*
specifies the SAS date, time, or datetime that specifies a project deadline. When this parameter is specified, the project is scheduled according to this project finish time.

_HEAD=*variable*
identifies the variable in the Activity data set that contains the name of the node on the head of an arc in the project network. This parameter is required when the project is in AOA (Activity-on-Arc) format.

_HOLIDUR=*variable*
identifies the variable in the Holiday data set that specifies the duration of the holiday. This variable is used to calculate the finish time of a holiday. If the _HOLIEND= parameter is specified, any value specified for the _HOLIDUR= parameter is ignored. This variable is optional; it has no default value.

_HOLIEND=*variable*
identifies the variable in the Holiday data set that specifies the finish time of each holiday. This variable is optional, and it has no default value.

_HOLISTART=*variable*
identifies the variable in the Holiday data set that specifies the start time of each non-workday. This variable is optional, and it has no default value.

_ID=*_id*
defines a string that lists the ID variables, separated by a space. The format of the _ID= parameter is `id1 id2 ...id`*n*, where *n* is the number of ID variables. When this parameter is specified, the macro passes ID variables to Microsoft Project. You can view them in Microsoft Project by inserting the columns ‘text1’ through ‘text*n*’.

_INTERVAL=*interval*
specifies the interval units by which task durations are measured. Possible values are DAY, WEEK, WEEKDAYS, WORKDAY, MONTH, QTR, YEAR, HOUR, MINUTE, SECOND, DTDAY, DTWRKDAY, DTWEEK, DTMONTH, DTQTR, DTYEAR, DTHOUR, DTMINUTE, and DTSECOND. The default value of _INTERVAL= is DAY.

_LAG=*_lag*
defines a string that lists the lag variables, separated by a space. The format of the _LAG= parameter is `lag1 lag2 ...lag`*n*, where *n* is the number of lag variables.

NOTE: Microsoft Project does not enable predecessors of a summary task to have a finish-to-finish or start-to-finish dependency.

NOTE: Microsoft Project does not permit the use of lag calendars. The %SASTOMSP macro converts calendar information. The names (or IDs) of lag calendars assigned to activities are saved in a text field in Microsoft Project called "Lag Calendars in SAS."

_PROJECT=*variable*
identifies the variable in the Activity data set that specifies the project to which an activity belongs. This parameter is required when tasks have hierarchical relationships.

_RESOBSTYPE=*variable*
identifies the character variable in the Resource data set that contains the type identifier for each observation. The _RESOBSTYPE= parameter is required if the RESDS= parameter is specified.

_RESOURCE=*_resource*
defines a string that lists the resource variables, separated by a space. The _RESOURCE= parameter is required if the RESDS= parameter is specified. The format of the *resource* parameter is `res1 res2 ...res`*n*, where *n* is the number of resource variables.

_RESPERIOD=*variable*
identifies the variable in the Resource data set that specifies the date from which a specified level of the resource is available for each observation with the _RESOBSTYPE= variable equal to 'RESLEVEL.'

_SUCCESSOR=*_successor*
defines a string that lists the successor variables in the Activity data set, separated by a space. The format of the _SUCCESSOR= parameter is `succ1 succ2 ...succ`*n*, where *n* is the number of successor variables. If this parameter is not specified, the macro identifies all variables in the Activity data set having names prefixed with 'SUCC' as successor variables.

_TAIL=*variable*
identifies the variable in the Activity data set that contains the name of the node on the tail of an arc in the project network. This parameter is required when the project is in AOA format.

UNIX Parameters

The following statements are available to establish a connection from SAS running on UNIX to a SAS PC Files Server. This enables you to run the %SASTOMSP macro on UNIX. For more information, see *SAS/ACCESS Interface to PC Files: Reference*.

DBMS=*identifier*
specifies the type of data to import. In this case, the value DBMS=accesscs must be specified.

PORT=*port-number*
specifies the port or service name that the SAS PC Files Server is listening to on the PC. The default value is 8621.

SERVER=*pc-server-hostname*
specifies the name of the computer on which you started the PC Files Server. This name is required by UNIX users to connect to this server machine and is reflected on the server control panel. This hostname can be specified as a simple computer name (e.g., wxp320), a fully qualified network name (e.g., `wxp320.domain.com`), or an IP address.

Default Values

Table 6.6 summarizes the list of parameters that have default values.

Table 6.6 Default Values

Parameter	Default Value
ACTDS	ACTIVITY
_ACTIVITY	ACTIVITY
_CALID	_CAL_
_DATE	SAS system date
_DAYLENGTH	`'8:00't`, when _INTERVAL is 'WORKDAY' or 'DTWRKDAY'; `'24:00't`, otherwise
_DAYSTART	`'9:00't`, when _INTERVAL is 'WORKDAY' or 'DTWRKDAY'; `'0:00't`, otherwise
_DUR	DUR
_INTERVAL	DAY
LIBRARY	'WORK' library
PROJ_NAME	'Project Imported from SAS'
_SUCCESSOR	List of all variables in Activity data set having names prefixed with 'SUCC'

Examples: The Microsoft Project Conversion Macros

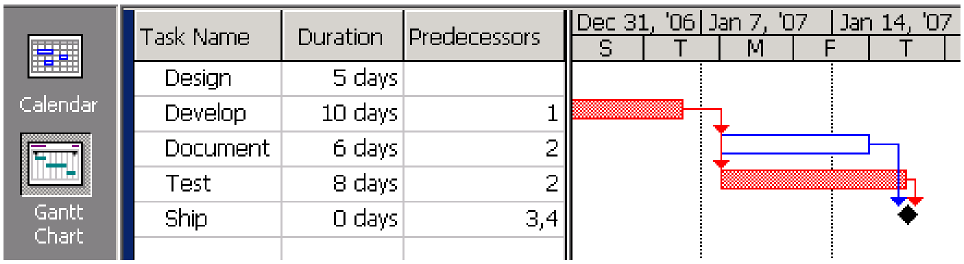
Since both the %MSPTOSAS and %SASTOMSP macros involve input and output activities, we assume for the sake of convenience that 'C:\MSPROJ\' is a valid path in the operating environment where the input or output files are located. You may need to replace the path appearing in the example programs before running them.

NOTE: Using Microsoft Access 2007 to open the provided Microsoft Project sample files might elicit the security warning 'Certain content in the database has been disabled'. The message can be safely ignored. The aforementioned sample files are named mspsas1.mdb, mspsas2.mdb, mspsas3.mdb, and mspsas4.mdb.

Example 6.1: Simple %MSPTOSAS Conversion

This example illustrates the use of the %MSPTOSAS macro. Consider the following project created in Microsoft Project 98, as shown in Output 6.1.1.

Output 6.1.1 Microsoft Project Window



You can use the following call to the %MSPTOSAS macro to convert the MS Project file and view it in the PM window:

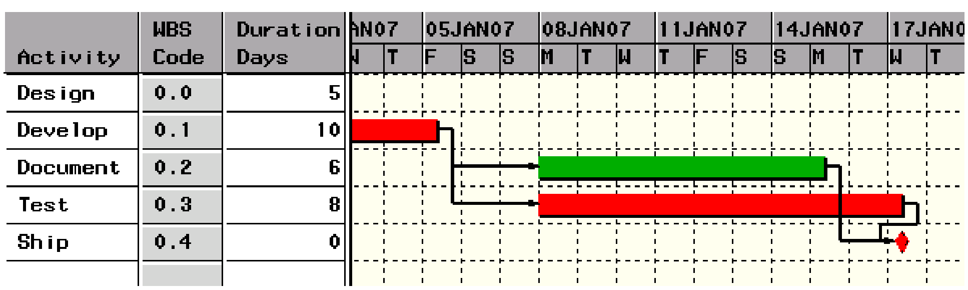
```
%msptosas(mdbfile=C:\MSPROJ\mspsas1.mdb, version=98)
```

Alternatively, you can specify the MDB file by using a file reference, as in the following example:

```
filename mspref "C:\MSPROJ\mspsas1.mdb";
%msptosas(mdbfile=mspref, version=98)
```

The PM window containing the preceding project is displayed in Output 6.1.2.

Output 6.1.2 PM Window



Because the LIBRARY= parameter is not specified, the default value 'WORK' is used. All eight data sets and the callpm.sas file are created in the 'WORK' library. The library reference 'mspout' is also assigned to this output library.

A partial view of the data set Activity generated during the conversion is shown in Output 6.1.3.

Output 6.1.3 Activity Data Set

ACTUID	SUCCUID	DURATIO	PNTUID	ACTIVITY	ALGNDAT	ALGNTYP	_CAL_
1000000	.	.	.	mspsas1	17DEC2006	SGE	1000001
1000001	.	5	1000000	Design	.		1000001
1000002	.	10	1000000	Develop	.		1000001
1000003	.	6	1000000	Document	.		1000001
1000004	.	8	1000000	Test	.		1000001
1000005	.	0	1000000	Ship	.		1000001
1000001	1000002	5	.		.		.
1000002	1000003	10	.		.		.
1000002	1000004	10	.		.		.
1000003	1000005	6	.		.		.
1000004	1000005	8	.		.		.

A partial view of the data set Schedule generated during the conversion is shown in Output 6.1.4.

Output 6.1.4 Schedule Data Set

Observation Type	Activity Variable	ACTIVITY	DURATION	Successo Variable	WBS Code for the Activity	ACTUID	Early Start	Early Finish
SCHEDULE	0	mspsas1	23	.	0	1000000	18DEC06:08:00:00	17JAN07:16:59:59
SCHEDULE	1	Design	5	.	0.0	1000001	18DEC06:08:00:00	22DEC06:16:59:59
SCHEDULE	2	Develop	10	.	0.1	1000002	25DEC06:08:00:00	05JAN07:16:59:59
SCHEDULE	3	Document	6	.	0.2	1000003	08JAN07:08:00:00	15JAN07:16:59:59
SCHEDULE	4	Test	8	.	0.3	1000004	08JAN07:08:00:00	17JAN07:16:59:59
SCHEDULE	5	Ship	0	.	0.4	1000005	17JAN07:16:59:59	17JAN07:16:59:59
LOGIC	1	Design	5	2		.	.	.
LOGIC	2	Develop	10	3		.	.	.
LOGIC	2	Develop	10	4		.	.	.
LOGIC	3	Document	6	5		.	.	.
LOGIC	4	Test	8	5		.	.	.

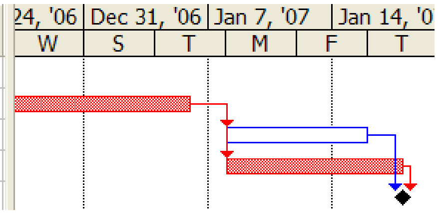
Note that in the Activity and Schedule data sets, there can be more than one observation associated with one activity. Some observations contain information about predecessor-successor relationships (LOGIC observations); the other observations contain the remaining activity information.

Example 6.2: Importing Activity Attributes

This example demonstrates the ability of the %MSPTOSAS macro to import Microsoft Project custom fields to a SAS data set. Consider the Microsoft Project window displayed in Output 6.2.1. The tasks and their precedence relationships are the same as in Example 6.1, but the version of the Microsoft Project is 2003 and there are two custom fields: “Flag1” and “Text1” (“Text1” is renamed as “Department Num.”)

Output 6.2.1 Microsoft Project Window

Task Name	Flag1	Department Num.	Duration
Design	Yes	R&D 5434	5 days
Develop	No	R&D 5305	10 days
Document	Yes	R&D 5313	6 days
Test	No	R&D 5444	8 days
Ship	No	R&D 5440	0 days



To convert this project, use the following SAS macro call:

```
%msptosas(mdbfile=C:\MSPROJ\mspsas2.mdb, version=2003)
```

After the conversion, the PM window is the same as in Output 6.1.2. However, in the resulting data set Task_Attributes there are two more variables: Flag1 and Text1 (Text1 has the label "Department Num."). Note that the three data sets Task_Attributes, Activity, and Schedule have two variables in common: ACTUID and ACTIVITY. ACTUID is numeric; ACTIVITY is character. Different activities can have identical ACTIVITY values, but each activity has a unique ACTUID value. Hence we recommend using ACTUID to identify the activities. For example, if you want to create a table consisting of the variables ACTIVITY and DURATION from the Activity data set, E_START and E_FINISH from the Schedule data set, and Flag1 and Text1 from the Task_Attributes data set, you can use the following SAS statements:

```
proc sql;
   create table merged as
      select a.ACTIVITY, a.DURATION,
         b.E_START, b.E_FINISH, c.Flag1, c.Text1
      from mspout.Activity a, mspout.Schedule b,
         mspout.Task_attributes c
      where a.ACTUID=b.ACTUID=c.ACTUID
         and a.ACTIVITY is not missing;
quit;
```

These statements merge variables from individual data sets. The variable ACTUID is used to identify activities; `a.ACTIVITY is not missing` is used to skip the logic observations.

The resulting data set Merged appears in Output 6.2.2.

Output 6.2.2 Merged Data Set

ACTIVITY	DURATION	Early Start	Early Finish	Flag1	Department Num.
mspsas2	.	18DEC06:08:00:00	17JAN07:16:59:59	NO	
Design	5	18DEC06:08:00:00	22DEC06:16:59:59	YES	R&D 5434
Develop	10	25DEC06:08:00:00	05JAN07:16:59:59	NO	R&D 5305
Document	6	08JAN07:08:00:00	15JAN07:16:59:59	YES	R&D 5313
Test	8	08JAN07:08:00:00	17JAN07:16:59:59	NO	R&D 5444
Ship	0	17JAN07:16:59:59	17JAN07:16:59:59	NO	R&D 5440

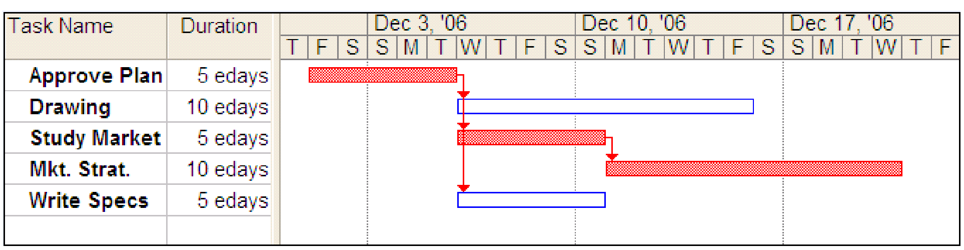
Note that a custom field is extracted and saved in the data set Task_Attributes only when the field has nondefault values for at least one task or when it has been renamed in Microsoft Project. If you are not sure whether or not Flag1 is present in the data set Task_attributes, you can use the following SAS statements instead:

```
%macro merge;
/* Open the data set mspout.Task_attributes and
   return an identifier &dsid. */
%let dsid=%sysfunc(open(mspout.Task_attributes));
proc sql;
     create table merged as
     select a.ACTIVITY, a.DURATION,
         b.E_START, b.E_FINISH,
         %if %sysfunc(varnum(&dsid,Flag1)) %then %do;
             c.Flag1,
         %end;
         c.Text1
     from mspout.Activity a, mspout.Schedule b,
         mspout.Task_attributes c
     where a.ACTUID=b.ACTUID=c.ACTUID
         and a.ACTIVITY is not missing;
quit;
/* Close the data set opened with the identifier &dsid. */
%let rc = %sysfunc (close (&dsid));
%mend;

%merge;
```

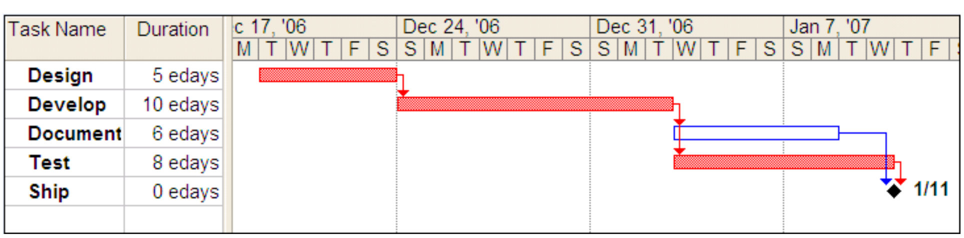
Example 6.3: Importing Multiple Projects

This example illustrates the ability of the %MSPTOSAS macro to convert multiple projects saved in one database file to a form readable by the PM procedure. Assume you have two projects, named “Software Project” and “Marketing Project,” in the Microsoft Access file mspsas3.mdb. The details of the projects are displayed in Output 6.3.1 and Output 6.3.2.

Output 6.3.1 Marketing Project: Microsoft Project Window



Output 6.3.2 Software Project: Microsoft Project Window

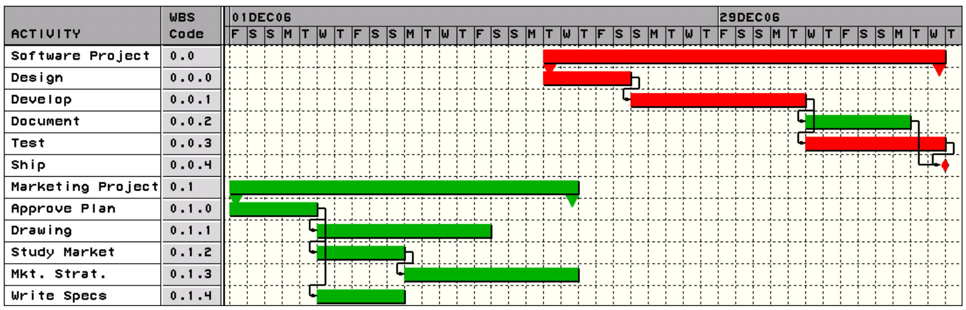


You can use the following call to the %MSPTOSAS macro to convert the projects and open them in the PM window:

```
%msptosas(mdbfile=C:\MSPROJ\mspsas3.mdb, version=2003)
```

The PM window containing the preceding projects is displayed in Output 6.3.3.

Output 6.3.3 PM Window



Example 6.4: Importing XML Files

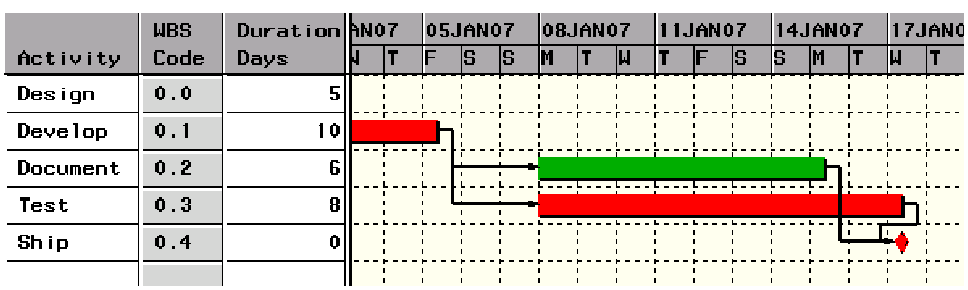
This example illustrates how to convert a Microsoft Project file that was saved in XML format to SAS data sets. Microsoft Project 2007 is initially used to save the project from Example 6.2 in XML format.

You can use the following SAS macro call to convert this project:

```
%msptosas(xmlfile=C:\MSPROJ\mspsas2.xml, version=2007)
```

Since the MAPFILE= option is not specified, the map file *mspxml.map* is created and used. After the conversion, the PM window that shows the converted project is displayed, as shown in Output 6.4.1. The display is the same as that shown in Output 6.1.2.You can customize the display by following the same steps as in Example 6.2. Output 6.4.2 shows the merged data set.

Output 6.4.1 PM Window



Output 6.4.2 Merged Data Set

ACTIVITY	DURATION	Early Start	Early Finish	Flag1	Department Num.
mspsas2	.	18DEC06:08:00:00	17JAN07:16:59:59	NO	
Design	5	18DEC06:08:00:00	22DEC06:16:59:59	YES	R&D 5434
Develop	10	25DEC06:08:00:00	05JAN07:16:59:59	NO	R&D 5305
Document	6	08JAN07:08:00:00	15JAN07:16:59:59	YES	R&D 5313
Test	8	08JAN07:08:00:00	17JAN07:16:59:59	NO	R&D 5444
Ship	0	17JAN07:16:59:59	17JAN07:16:59:59	NO	R&D 5440

Example 6.5: Simple %SASTOMSP Conversion

This example demonstrates how to convert a simple project from SAS to Microsoft Project. The data set Activity is created using the following SAS DATA step:

```
data activity;
   format activity succ1 $8.;
   input activity dur succ1;
   datalines;
Design     5   Develop
Develop   10   Document
Develop   10   Test
Document   6   Ship
Test       8   Ship
Ship       0   .
;
```

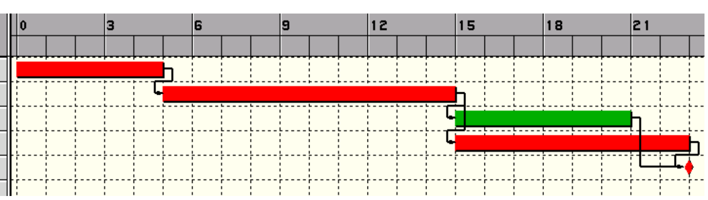
You can display this project in the PM procedure with the following SAS statements:

```
proc pm data=activity;
    act activity;
    succ succ1;
    duration dur;
run;
```

The resulting PM window is shown in Output 6.5.1. Note that the project starts at time 0, and there are no time units.

Output 6.5.1 PM Window

activity	Duration	Early Start
Design	5	0
Develop	10	5
Document	6	15
Test	8	15
Ship	0	23

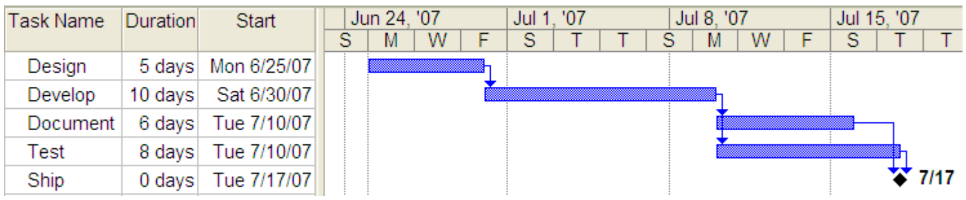


To convert the project specified by a SAS data set to an MDB file that is readable by Microsoft Project, you use the %SASTOMSP macro. You need to use the MDBFILE= parameter to specify the location and name of the MDB file to be created. For example, the following statement converts the project specified by the SAS data set Activity to the MDB file ‘C:\MSPROJ\sasmsp1a.mdb’:

```
filename mspref "C:\MSPROJ\sasmsp1a.mdb";
%sastomsp(mdbfile=mspref);
```

Note that several parameters are omitted since the variable names in the data set Activity are identical to the corresponding default parameters of the conversion macro (see the section "Default Values" on page 366). Output 6.5.2 shows the resulting project schedule as viewed in Microsoft Project.

Output 6.5.2 Microsoft Project Window



The schedule seen in Output 6.5.1 (the PM window) is represented in terms of the time interval (days) while that in Output 6.5.2 (the MS Project window) has dates. The PM procedure does not use dates if none are specified, so the project is scheduled to begin at time 0 and end at time 23. However, Microsoft Project schedules projects by using dates. Since no start date is specified, the conversion macro uses the current day as the default starting date.

To create a project schedule that is consistent across both the PM procedure and Microsoft Project, you need to add an additional option in both the PM invocation and the %SASTOMSP macro statement. In the PM invocation, you need to add the DATE= option to start the project on a given date. In the following example, the project is started on December 15, 2006:

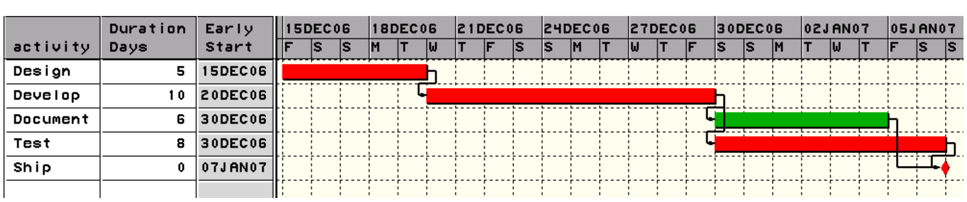
```
proc pm data=activity date='15Dec06'd;
   act activity;
   succ succ1;
   duration dur;
run;
```

For the call to %SASTOMSP, you specify this same starting date by using the _DATE= parameter:

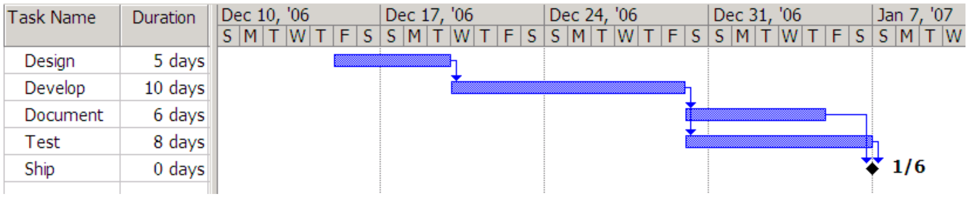
```
filename mspref "C:\MSPROJ\sasmsp1b.mdb";
%sastomsp(mdbfile=mspref, _date='15Dec06'd);
```

The resulting windows are shown in Output 6.5.3 and Output 6.5.4.

Output 6.5.3 PM Window



Output 6.5.4 Microsoft Project Window



Example 6.6: Exporting Data Set and Variable Names

In this example, the same project from Example 6.5 is used. However, in this case, the data set names and variable names do not have the default variables, so they must be explicitly specified in the %SASTOMSP macro.

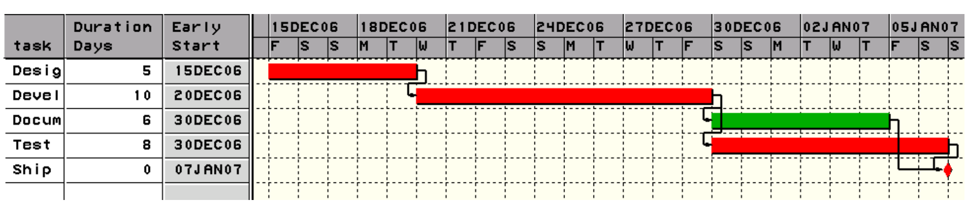
For example, suppose the data set containing the activity information is created as follows:

```
data software;
   format task s1 s2 $8.;
   input task duration s1 s2;
   datalines;
Design     5  Develop   .
Develop   10  Document  Test
Document   6  Ship      .
Test       8  Ship      .
Ship       0  .         .
;
```

This data set is identical to the data set Activity in Example 6.5, except for a slightly different format. The following PM invocation results in the PM window shown in Output 6.6.1:

```
proc pm data=software date='15Dec06'd;
   act task;
   succ s1 s2;
   duration duration;
run;
```

Output 6.6.1 PM Window



To convert the data to an MDB file, you need to specify several additional parameters in the %SASTOMSP macro call. First, you need to specify the name of the data set (Software in this case) using the ACTDS= parameter. In Example 6.1, this parameter was not needed, because the default data set name Activity was used.

Similarly, you must specify the names of the activity, duration, and successor variables using the _ACTIVITY=, _DUR=, and _SUCCESSOR= parameters, respectively.

The following call to the %SASTOMSP macro results in the Microsoft Project window as shown in Output 6.5.4:

```
filename mspref "C:\MSPROJ\sasmsp2.mdb";
%sastomsp(mdbfile=mspref,
          actds=software,
          _activity=task,
          _dur=duration,
          _successor=s1 s2,
          _date='15Dec06'd);
```

Example 6.7: Exporting Calendars and Holidays

This example demonstrates the capability of the %SASTOMSP macro to handle multiple calendars within a project. Each activity is associated with one of four available calendars. Each calendar is customized to incorporate various workday shift patterns. In addition, there is a holiday data set that is also appropriately associated with a calendar. The four different calendars are defined as follows:

- The DEFAULT calendar has five eight-hour days (Monday through Friday) and holidays on Saturday and Sunday.
- The calendar OVT_CAL specifies an overtime calendar that has 10-hour workdays on Monday through Friday, a half-day on Saturday, and a holiday on Sunday.
- The calendar PROD_CAL follows a more complicated work pattern: Sunday is a holiday; on Monday work is done from 8 a.m. through midnight with a two hour break from 6 p.m. to 8 p.m.; on Tuesday through Friday work is done round the clock with two 2-hour breaks, one from 6 a.m to 8 a.m and the other from 6 p.m. to 8 p.m.; on Saturday the work shifts are from midnight to 6 a.m. and again from 8 a.m. to 6 p.m.
- The calendar ENG_CAL is specified similar to the default calendar, but with an extra vacation period of 7 days beginning on December 8.

The following data set contains the activity information:

```
data actdat;
   format activities $12. s1-s3 $12. cal $8.;
   input activities & days s1 & s2 & s3 & cal &;
   datalines;
Approve Plan   5  Drawings      Study Market  Write Specs  DEFAULT
Drawings      10  Prototype     .             .            DEFAULT
Study Market   5  Mkt. Strat.   .             .            DEFAULT
Write Specs    5  Prototype     .             .            ENG_CAL
Prototype     15  Materials     Facility      .            OVT_CAL
Mkt. Strat.   10  Test Market   Marketing     .            DEFAULT
Materials     10  Init. Prod.   .             .            DEFAULT
Facility      10  Init. Prod.   .             .            DEFAULT
Init. Prod.   10  Test Market   Marketing     Evaluate     DEFAULT
Evaluate      10  Changes       .             .            DEFAULT
Test Market   15  Changes       .             .            DEFAULT
```

```
Changes        5  Production    .             .             DEFAULT
Production     0  .             .             .             PROD_CAL
Marketing      0  .             .             .             DEFAULT
;
```

The next three data sets specify the work shift patterns, calendars, and holidays, respectively:

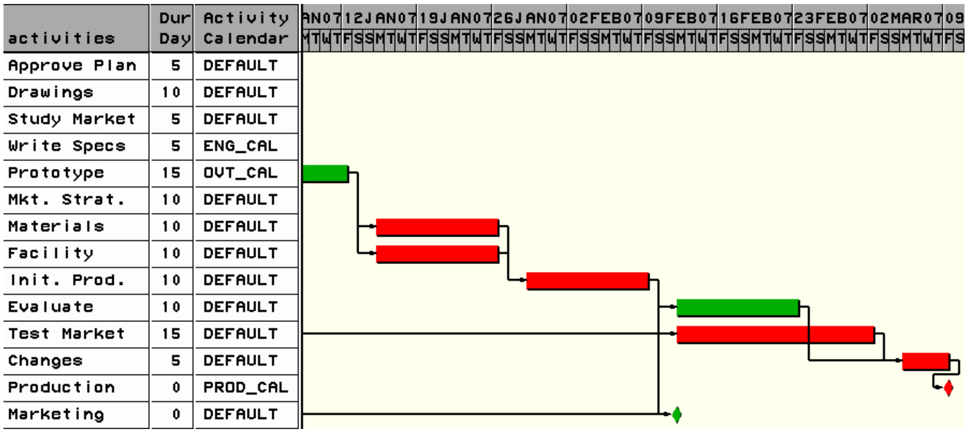
```
data wrkdat;
   input fullday time8. halfday time8. ovtday time8.
         d1 time8. d2 time8. d3 time8.;
   format fullday halfday ovtday d1 d2 d3 time6.;
   datalines;
08:00   08:00   08:00   .       08:00   .
16:00   12:00   18:00   06:00   18:00   06:00
.       .       .       08:00   20:00   08:00
.       .       .       18:00   .       18:00
.       .       .       20:00   .       .
.       .       .       .       .       .
;

data caldat;
   input  cal $ _sun_ $ _mon_$ _tue_$  _wed_$ _thu_$ _fri_$ _sat_ $;
   datalines;
DEFAULT  holiday fullday fullday fullday fullday fullday holiday
OVT_CAL  holiday ovtday  ovtday  ovtday  ovtday  ovtday  halfday
PROD_CAL holiday d2      d1      d1      d1      d1      d3
ENG_CAL  .       .       .       .       .       .       .
;

data holdat;
   format holiday holifin date7.;
   input holiday & date7. holifin & date7. holidur cal $;
   datalines;
08Dec06  .        7  ENG_CAL
24Dec06  26Dec06  .  .
01Jan07  01Jan07  .  .
;
```

You can view the resulting project schedule in the PM window with the following call:

```
proc pm date='01Dec06'd data=actdat
    calendar=caldat holidata=holdat
    workday=wrkdat
    daylength='08:00't;
    activity activities;
    duration days;
    successor s1 s2 s3;
    calid cal;
    holiday holiday / holifin = holifin holidur=holidur;
run;
```

The resulting PM window is shown in Output 6.7.1.

Output 6.7.1 PM Window

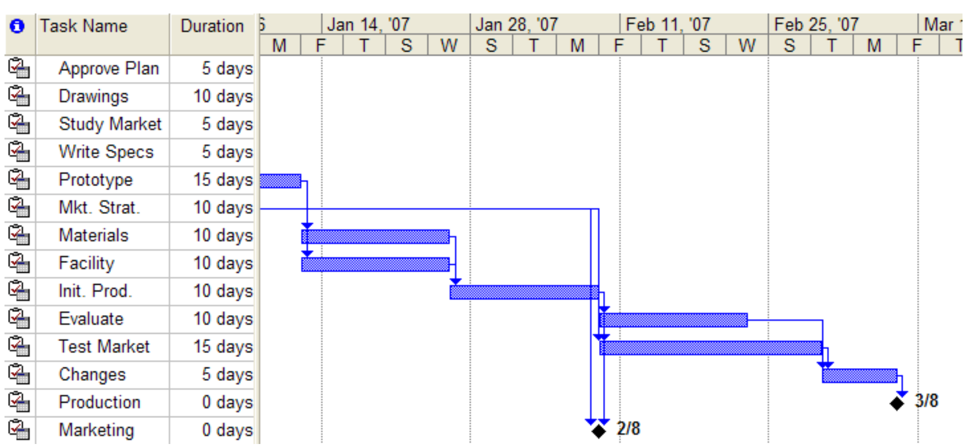


To convert the preceding data to an MDB file that includes the calendar and holiday information, you can use the following call to %SASTOMSP:

```
%sastomsp(mdbfile=mspref,
          actds=actdat, calds=caldat,
          holds=holdat, workds=wrkdat,
          _date='1Dec06'd,
          _daylength='08:00't,
          _activity=activities,
          _dur=days,
          _successor=s1 s2 s3,
          _calid=cal,
          _holistart=holiday, _holiend=holifin, _holidur=holidur)
```

The resulting MS Project Window is shown in Output 6.7.2. The schedule is the same as the one produced by the PM procedure.

Output 6.7.2 MS Project Window



Example 6.8: Exporting Resource-Constrained Schedules

In this example, a project is scheduled subject to resource constraints.

Before you continue, make sure that in your Microsoft Project software, “resource leveling” is properly set as follows so that the resource-constrained schedule is automatically displayed in the Gantt chart. From the Microsoft Project **Tools** menu, select **Level Resources** (**Resource Leveling** in Microsoft Project 2000). Then select **Automatic** for **Leveling calculations**. Clear **Level only within available slack** for **Resolving overallocations**. Then click on **OK**.

The following DATA steps specify the project in this example. Resource assignments are specified in the data set Activity, and the resource availabilities are defined in the data set Resources.

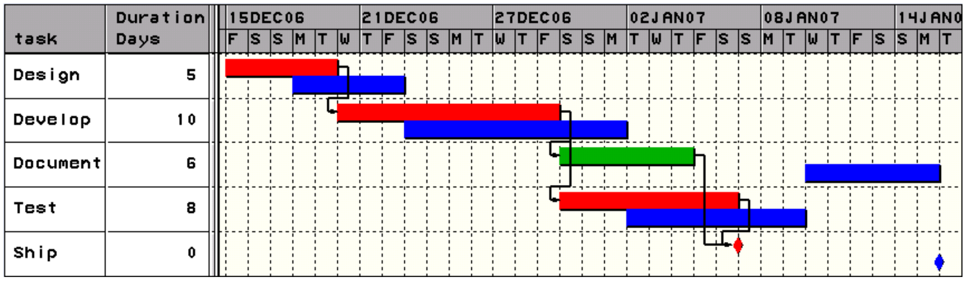
```
data activity;
   format task succ1 $8.;
   input task dur succ1 engineer writer tester;
   datalines;
Design     5  Develop   1  .  1
Develop   10  Document  1  .  1
Develop   10  Test      1  .  1
Document   6  Ship      1  1  .
Test       8  Ship      1  .  1
Ship       0  .         .  .  .
;

data resources;
   format obstype $8.;
   input obstype date: date7. engineer writer tester;
   datalines;
reslevel  15Dec06  .  .  1
reslevel  18Dec06  1  .  .
reslevel  30Dec06  .  1  .
;
```

The PM invocation is given in the following statements. The resource data set is specified with `resin=resources;` and the RESOURCE statement is also added to identify the applicable resources.

```
proc pm data=activity
   date='15Dec06'd
   resin=resources;
   act task;
   succ succ1;
   duration dur;
   resource engineer writer tester / period=date;
run;
```

The resulting PM window is shown in Output 6.8.1. Notice that both the early schedule and the resource-constrained schedule are displayed in the Gantt chart.

Output 6.8.1 PM Window

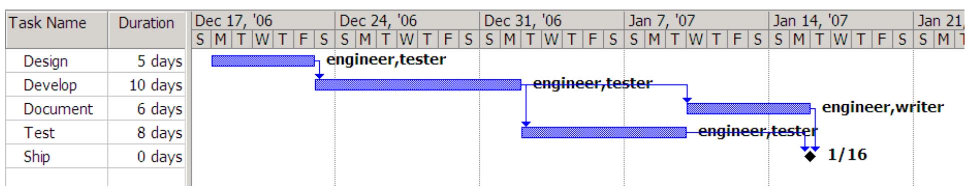


To convert the data to an MDB file that includes resource requirements, you use the following call to %SASTOMSP:

```
%sastomsp(mdbfile=mspref,
          resds=resources,
          _activity=task,
          _date='15Dec06'd,
          _resobstype=obstype,
          _resource=engineer writer tester,
          _resperiod=date);
```

The resulting MS Project window is shown in Output 6.8.2.

Output 6.8.2 MS Project Window



In Output 6.8.2, Microsoft Project displays only the resource-constrained schedule. To get a comparison view as in the PM procedure, you can save the output schedule of the PM procedure and specify the SCHEDULEDS= parameter in the call to %SASTOMSP, as follows:

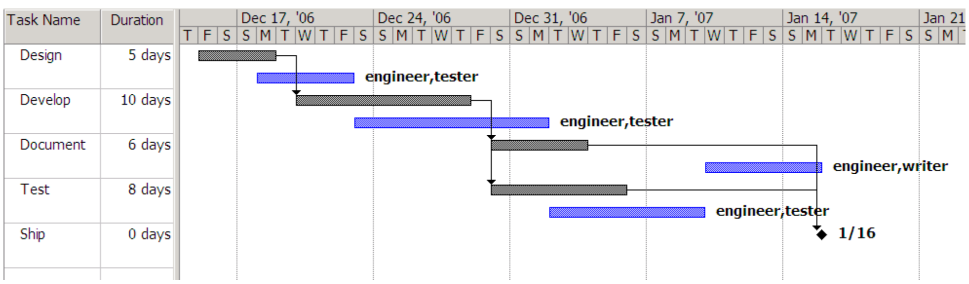
```
proc pm data=activity
   date='15Dec06'd
   resin=resources out=schedule;
   act task;
   succ succ1;
   duration dur;
   resource engineer writer tester / period=date;
run;

filename mspref "C:\MSPROJ\sasmsp4b.mdb";
%sastomsp(mdbfile=mspref,
          resds=resources,
          scheduleds=schedule,
          _activity=task,
          _date='15Dec06'd,
          _resobstype=obstype,
```

```
               _resource=engineer writer tester,
               _resperiod=date);
```

The PROC PM window is identical to the one shown in Output 6.8.1, and the resulting Microsoft Project window is identical to the one shown in Output 6.8.2. However, the schedule information computed by SAS has been exported to the MDB file and is available for display in Microsoft Project.

You can manually set Microsoft Project to display the schedules in the Gantt Chart. In Microsoft Project, from the **Format** menu, select **Bar Styles**. Then change the **Task** number in the column **Row** from **1** to **2**. Click on **Insert Row**. Type a name for the new row (e.g., SAS Early Schedule). Under **From**, select **Start1 (SAS_E_start)**. Under **To**, select **Finish1 (SAS_E_finish)**. Click **OK**. The resulting window is shown in Output 6.8.3. Now both the resource-constrained schedule and the early schedule are displayed in the Gantt chart.

Output 6.8.3 MS Project Window

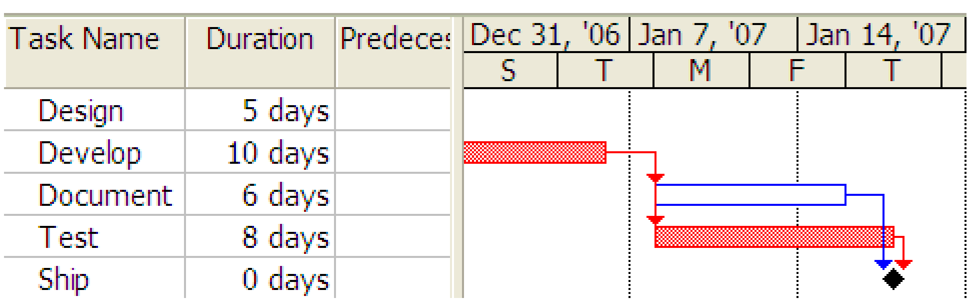


You can also display the SAS_E_start and the SAS_E_finish in columns by going to the **Insert** menu, selecting **Column**, and selecting the **Start1 (SAS_E_start)** and **Finish1 (SAS_E_finish)** fields.

Example 6.9: Round Trip between a SAS Program and Microsoft Project

This example demonstrates how to convert a Microsoft Project (MSP) database file into SAS by using the %MSPTOSAS macro, and then to convert the file back to MSP by using the %SASTOMSP macro.

Output 6.9.1 MS Project Window



Suppose you want to convert the MSP project shown in Output 6.9.1 into SAS software. You can convert the corresponding MDB file into a SAS data set by using the %MSPTOSAS macro, as follows:

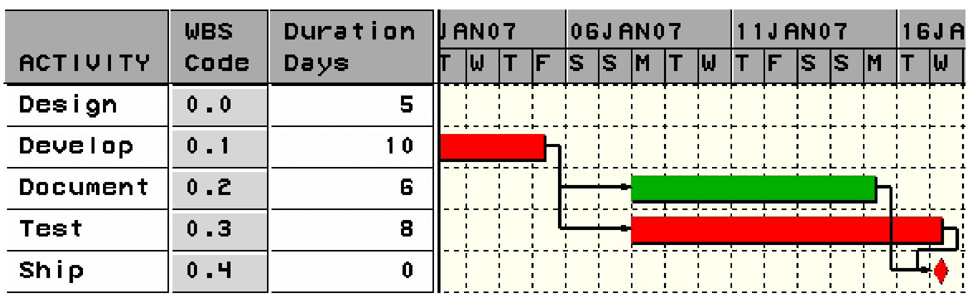
```
%msptosas(mdbfile=C:\MSPROJ\mspsas4.mdb, library=C:\MSPROJ,
          version=2003)
```

The %MSPTOSAS macro generates the data sets Activity, Calendar, Holiday, Workday, Resource, Schedule, Task_Attributes, and Prefs, as well as the file callpm.sas. The following SAS statements can be found either in the callpm.sas file or in the SAS log:

```
libname mspout "C:\MSPROJ";
PROC PM data = mspout.activity project=mspout.prefs
     caledata = mspout.calendar
     workdata = mspout.workday
     out=mspout.schedule
     interval=dtday
     date="17DEC06:08:00:00"dt
     daylength=" 8:00"t
     suppressobswarn
     setfinishmilestone;
activity ACTID;
successor SUCCUID / LAG = LAG;
duration DURATION;
project PNTUID;
id ACTIVITY ACTUID;
RUN;
```

Output 6.9.2 shows the resulting PM window.

Output 6.9.2 PM Window



By taking the parameters from the preceding PM invocation, you can specify the values in the following %SASTOMSP call. The %SASTOMSP macro converts the project back into Microsoft Project format.

```
filename mspref "C:\MSPROJ\sasmsp5.mdb";
%sastomsp(library=C:\MSPROJ, mdbfile=mspref,
          actds=activity, calds=calendar, workds=workday,
          scheduleds=schedule, _interval=dtday,
          _date="17DEC06:08:00:00"dt, _daylength=" 8:00"t,
          _activity=ACTUID, _successor=SUCCUID,
          _lag=LAG, _dur=DURATION, _project=PNTUID,
          _id=ACTIVITY ACTUID)
```

In this example, after the project was converted to SAS code, the same project was converted back to Microsoft Project. The round trip between Microsoft Project and SAS software enables you to harness the power of SAS programming in scheduling, resource leveling, data processing, and more. Once your calculations are complete, you can return the results to Microsoft Project. Note that because SAS/OR Project Management and Microsoft Project use different strategies in calculating schedules, your project might show different start or finish times in SAS software and Microsoft Project. To compare the differences, you can display SAS schedules both in a table column view and in a Gantt chart, as described in Example 6.8.

Chapter 7
The DTREE Procedure

Contents

Overview: DTREE Procedure

The DTREE procedure in SAS/OR software is an interactive procedure for decision analysis. The procedure interprets a decision problem represented in SAS data sets, finds the optimal decisions, and plots on a line printer or a graphics device the decision tree showing the optimal decisions.

To use PROC DTREE you first construct a decision model to represent your problem. This model, called *a generic decision tree model*, is made up of *stages*.[1] Every stage has a *stage name*, which identifies the stage, as well as a *type*, which specifies the type of the stage. There are three types of stages: decision stages, chance stages, and end stages. In addition, every stage has possible *outcomes*.

A *decision stage* represents a particular decision you have to make. The outcomes of a decision stage are the possible *alternatives* (or *actions*) of the decision. A *chance stage* represents an *uncertain factor* in the decision problem (a statistician might call it a *random variable*; here it is called an *uncertainty*). The outcomes of a chance stage are *events*, one of which will occur according to a given probability distribution. An *end stage* terminates a particular *scenario* (a sequence of alternatives and events). It is not necessary to include an end stage in your model; the DTREE procedure adds an end stage to your model if one is needed.

Each outcome of a decision or chance stage also has several attributes, an *outcome name* to identify the outcome, a *reward* to give the instant reward of the outcome, and a *successor* to specify the name of the stage that comes next when this outcome is realized. For chance stages, a *probability* attribute is also needed. It gives the relative likelihood of this outcome. Every decision stage should have at least two alternatives, and every chance stage should have at least two events. Probabilities of events for a chance stage *must* sum to 1. End stages do not have any outcomes.

The structure of a decision model is given in the STAGEIN= data set. It contains the stage name, the type, and the attributes (except probability) of all outcomes for each stage in your model. You can specify each stage in one observation or across several observations. If a diagrammatic representation of a decision problem is all you want, you probably do not need any other data sets.

If you want to evaluate and analyze your decision problem, you need another SAS data set, called the PROBIN= data set. This data set describes the probabilities or conditional probabilities for every event in your model. Each observation in the data set contains a list of given conditions (list of outcomes), if there are any, and at least one combination of event and probability. Each event and probability combination identifies

[1]The stages are often referred to as *variables* in many decision analysis articles.

the probability that the event occurs given that all the outcomes specified in the list occur. If no conditions are given, then the probabilities are unconditional probabilities.

The third data set, called the PAYOFFS= data set, contains the value of each possible scenario. You can specify one or more scenarios and the associated values in one observation. If the PAYOFFS= data set is omitted, the DTREE procedure assumes that all values are zero and uses rewards for outcomes to evaluate the decision problem.

You can use PROC DTREE to display, evaluate, and analyze your decision problem. In the PROC DTREE statement, you specify input data sets and other options. A VARIABLES statement identifies the variables in the input data set that describe the model. This statement can be used only once and must appear immediately after the PROC DTREE statement. The EVALUATE statement evaluates the decision tree. You can display the optimal decisions by using the SUMMARY statement, or you can plot the complete tree with the TREEPLOT statement. Finally, you can also associate HTML pages with decision tree nodes and create Web-enabled decision tree diagrams.

It is also possible to interactively modify some attributes of your model with the MODIFY statement and to change the order of decisions by using the MOVE statement. Before making any changes to the model, you should save the current model with the SAVE statement so that you can call it back later by using the RECALL statement. Questions about the value of perfect information or the value of perfect control are answered using the VPI and VPC statements. Moreover, any options that can be specified in the PROC DTREE statement can be reset at any time with the RESET statement.

All statements can appear in any order and can be used as many times as desired with one exception. The RECALL statement must be preceded by at least one SAVE statement. In addition, only one model can be saved at any time; the SAVE statement overwrites the previously saved model. Finally, you can use the QUIT statement to stop processing and exit the procedure.

The DTREE procedure produces one output data set. The IMAGEMAP= data set contains the outline coordinates for the nodes in the decision tree that can be used to generate HTML MAP tags.

PROC DTREE uses the Output Delivery System (ODS), a SAS subsystem that provides capabilities for displaying and controlling the output from SAS procedures. ODS enables you to convert any of the output from PROC DTREE into a SAS data set. For further details, refer to the chapter on ODS in the *SAS/STAT User's Guide*.

Getting Started: DTREE Procedure

Introductory Example

A decision problem for an oil wildcatter illustrates the use of the DTREE procedure. The oil wildcatter must decide whether or not to drill at a given site before his option expires. He is uncertain about many things: the cost of drilling, the extent of the oil or gas deposits at the site, and so on. Based on the reports of his technical staff, the hole could be `'Dry'` with probability 0.5, `'Wet'` with probability 0.3, and `'Soaking'` with probability 0.2. His monetary payoffs are given in the following table.

Table 7.1 Monetary Payoffs of Oil Wildcatter's Problem

	Drill	Not Drill
Dry	0	0
Wet	$700,000	0
Soaking	$1,200,000	0

The wildcatter also learned from the reports that the cost of drilling could be $150,000 with probability 0.2, $300,000 with probability 0.6, and $500,000 with probability 0.2. He can gain further relevant information about the underlying geological structure of this site by conducting seismic soundings. A cost control procedure that can make the probabilities of the `'High'` cost outcomes smaller (and hence, the probabilities of the `'Low'` cost outcomes larger) is also available. However, such information and control are quite costly, about $60,000 and $120,000, respectively. The wildcatter must also decide whether or not to take the sounding test or the cost control program before he makes his final decision: to drill or not to drill.

The oil wildcatter feels that he should structure and analyze his basic problem first: whether or not to drill. He builds a model that contains one decision stage named `'Drill'` (with two outcomes, `'Drill'` and `'Not_Drill'`) and two chance stages named `'Cost'` and `'Oil_Deposit'`. A representation of the model is saved in three SAS data sets. In particular, the STAGEIN= data set can be saved as follows:

```
/* -- create the STAGEIN= data set             -- */
data Dtoils1;
format _STNAME_ $12. _STTYPE_ $2. _OUTCOM_ $10.
     _SUCCES_ $12. ;
input _STNAME_ $ _STTYPE_ $ _OUTCOM_ $ _SUCCES_ $ ;
datalines;
Drill        D   Drill        Cost
.            .   Not_Drill    .
Cost         C   Low          Oil_Deposit
.            .   Fair         Oil_Deposit
.            .   High         Oil_Deposit
Oil_Deposit C    Dry          .
.            .   Wet          .
.            .   Soaking      .
;
```

The structure of the decision problem is given in the Dtoils1 data set. As you apply this data set, you should be aware of the following points:

- There is no reward variable in this data set; it is not necessary.
- The ordering of the chance stages `'Cost'` and `'Oil_Deposit'` is arbitrary.
- Missing values for the _SUCCES_ variable are treated as `'_ENDST_'` (the default name of the end stage) unless the associated outcome variable (_OUTCOM_) is also missing.

The following PROBIN= data set contains the probabilities of events:

```
/* -- create the PROBIN= data set                -- */
data Dtoilp1;
input _EVENT1 $ _PROB1
      _EVENT2 $ _PROB2
      _EVENT3 $ _PROB3 ;
datalines;
Low   0.2   Fair   0.6   High     0.2
Dry   0.5   Wet    0.3   Soaking  0.2
;
```

Notice that the sum of the probabilities of the events `'Low'`, `'Fair'`, and `'High'` is 1.0. Similarly, the sum of the probabilities of the events `'Dry'`, `'Wet'`, and `'Soaking'` is 1.0.

Finally, the following statements produce the PAYOFFS= data set that lists all possible scenarios and their associated payoffs.

```
/* -- create PAYOFFS= data set                  -- */
data Dtoilu1;
format _STATE1-_STATE3 $12. _VALUE_ dollar12.0;
input _STATE1 $ _STATE2 $ _STATE3 $ ;

/* determine the cost for this scenario */
if _STATE1='Low' then _COST_=150000;
else if _STATE1='Fair' then _COST_=300000;
else _COST_=500000;

/* determine the oil deposit and the          */
/* corresponding net payoff for this scenario */
if _STATE2='Dry' then _PAYOFF_=0;
else if _STATE2='Wet' then _PAYOFF_=700000;
else _PAYOFF_=1200000;

/* calculate the net return for this scenario */
if _STATE3='Not_Drill' then _VALUE_=0;
else _VALUE_=_PAYOFF_-_COST_;

/* drop unneeded variables */
drop _COST_ _PAYOFF_;

datalines;
Low         Dry         Not_Drill
Low         Dry         Drill
Low         Wet         Not_Drill
Low         Wet         Drill
Low         Soaking     Not_Drill
Low         Soaking     Drill
Fair        Dry         Not_Drill
Fair        Dry         Drill
Fair        Wet         Not_Drill
Fair        Wet         Drill
Fair        Soaking     Not_Drill
Fair        Soaking     Drill
```

```
High        Dry         Not_Drill
High        Dry         Drill
High        Wet         Not_Drill
High        Wet         Drill
High        Soaking     Not_Drill
High        Soaking     Drill
;
```

This data set can be displayed, as shown in Figure 7.1, with the following PROC PRINT statements:

```
/* -- print the payoff table                    -- */

title "Oil Wildcatter's Problem";
title3 "The Payoffs";

proc print data=Dtoilu1;
run;
```

Figure 7.1 Payoffs of the Oil Wildcatter's Problem

Oil Wildcatter's Problem

The Payoffs

Obs	_STATE1	_STATE2	_STATE3	_VALUE_
1	Low	Dry	Not_Drill	$0
2	Low	Dry	Drill	$-150,000
3	Low	Wet	Not_Drill	$0
4	Low	Wet	Drill	$550,000
5	Low	Soaking	Not_Drill	$0
6	Low	Soaking	Drill	$1,050,000
7	Fair	Dry	Not_Drill	$0
8	Fair	Dry	Drill	$-300,000
9	Fair	Wet	Not_Drill	$0
10	Fair	Wet	Drill	$400,000
11	Fair	Soaking	Not_Drill	$0
12	Fair	Soaking	Drill	$900,000
13	High	Dry	Not_Drill	$0
14	High	Dry	Drill	$-500,000
15	High	Wet	Not_Drill	$0
16	High	Wet	Drill	$200,000
17	High	Soaking	Not_Drill	$0
18	High	Soaking	Drill	$700,000

The $550,000 payoff associated with the scenario `'Low'`, `'Wet'`, and `'Drill'` is a net figure; it represents a return of $700,000 for a wet hole less the $150,000 cost for drilling. Similarly, the net return of the consequence associated with the scenario `'High'`, `'Soaking'`, and `'Drill'` is $700,000, which is interpreted as a return of $1,200,000 less the $500,000 `'High'` cost.

Now the wildcatter can invoke PROC DTREE to evaluate his model and to find the optimal decision using the following statements:

```
   /* -- PROC DTREE statements                    -- */
title "Oil Wildcatter's Problem";

proc dtree stagein=Dtoils1
           probin=Dtoilp1
           payoffs=Dtoilu1
           nowarning;

   evaluate / summary;
```

The following message, which notes the order of the stages, appears on the SAS log:

```
NOTE: Present order of stages:

      Drill(D), Cost(C), Oil_Deposit(C), _ENDST_(E).
```

Figure 7.2 Optimal Decision Summary of the Oil Wildcatter's Problem

Oil Wildcatter's Problem

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Drill	Decision
Cost	Chance
Oil_Deposit	Chance
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Expected Value (MAXEV)
Optimal Decision Yields:	$140,000

Optimal Decision Policy Up to Stage Drill		
Alternatives or Outcomes	**Cumulative Reward**	**Evaluating Value**
Drill		$140,000*
Not_Drill		$0

The SUMMARY option in the EVALUATE statement produces the optimal decision summary shown in Figure 7.2.

The summary shows that the best action, in the sense of maximizing the expected payoff, is *to drill*. The expected payoff for this optimal decision is $140,000, as shown on the summary.

Perhaps the best way to view the details of the results is to display the complete decision tree. The following statement draws the decision tree, as shown in Figure 7.3, in line-printer format:

```
   /* plot decision tree diagram in line-printer mode */
OPTIONS LINESIZE=100;
treeplot/ lineprinter;
```

Figure 7.3 The Decision Tree

Oil Wildcatter's Problem

The DTREE Procedure

Line-printer Plot

```
                                                                               Dry
                                                                           -----------------------E
                                                                           | p=0.5  EV=  $-150,000
                                                     Low                   |   Wet
                                                  -----------------------C-|----------------------E
                                                  | p=0.2  EV=   $300,000  | p=0.3  EV=   $550,000
                                                  |                        | Soaking
                                                  |                        -----------------------E
                                                  |                          p=0.2  EV= $1,050,000
                                                  |                            Dry
                                                  |                        -----------------------E
                                                  |                        | p=0.5  EV=  $-300,000
                           Drill                  | Fair                   |   Wet
                        -=======================C-|----------------------C-|----------------------E
                        |        EV=   $140,000   | p=0.6  EV=   $150,000  | p=0.3  EV=   $400,000
                        |                         |                        | Soaking
                        |                         |                        -----------------------E
                        |                         |                          p=0.2  EV=   $900,000
                        |                         |                            Dry
----------------------D-|                         |                        -----------------------E
        EV=   $140,000  |                         |                        | p=0.5  EV=  $-500,000
                        |                         | High                   |   Wet
                        |                         -----------------------C-|----------------------E
                        |                           p=0.2  EV=  $-50,000  | p=0.3  EV=   $200,000
                        |                                                  | Soaking
                        |                                                  -----------------------E
                        |                                                    p=0.2  EV=   $700,000
                        |Not_Drill
                        ------------------------E
                                 EV=         $0
```

Attitudes toward Risk

Assume now that the oil wildcatter is constantly risk averse and has an exponential utility function with a *risk tolerance* (RT) of $700,000. The risk tolerance is a measure of the decision maker's attitude to risk. See the section "Evaluation" on page 425 for descriptions of the utility function and risk tolerance.

The new optimal decision based on this utility function can be determined with the following statement:

```
evaluate / criterion=maxce rt=700000 summary;
```

The summary, shown in Figure 7.4, indicates that the venture of investing in the oil well is worth $-13,580 to the wildcatter, and he should not drill the well.

Figure 7.4 Summary of the Oil Wildcatter's Problem with RT = $700,000

Oil Wildcatter's Problem

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Drill	Decision
Cost	Chance
Oil_Deposit	Chance
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Certain Equivalent Value (MAXCE)
Risk Tolerance:	$700,000
Optimal Decision Yields:	$0

Optimal Decision Policy
Up to Stage Drill

Alternatives or Outcomes	Cumulative Reward	Evaluating Value
Drill		$-13,580
Not_Drill		$0*

Sensitivity Analysis and Value of Perfect Information

The oil wildcatter learned that the optimal decision changed when his attitude toward risk changed. Since risk attitude is difficult to express quantitatively, the oil wildcatter wanted to learn more about the uncertainties in his problem. Before spending any money on information-gathering procedures, he would like to know the benefit of knowing, before the `'Drill'` or `'Not_Drill'` decision, the amount of oil or the cost of drilling. The simplest approach is to calculate the value of perfect information for each uncertainty. This quantity gives an upper limit on the amount that could be spent profitably on information gathering. The expected value of information for the amount of oil is calculated by the following statement:

```
vpi Oil_Deposit;
```

The result of the previous statement is written to the SAS log as

```
NOTE: The currently optimal decision yields 140000.
```

```
NOTE: The new optimal decision yields 295000.
NOTE: The value of perfect information of stage Oil_Deposit
      yields 155000.
```

This means that the wildcatter could spend up to $155,000 to determine the amount of oil in the deposit with certainty before losing money. There are several alternative ways to calculate the expected value of perfect information. For example, the following statement

```
vpi Cost;
```

is equivalent to

```
save;
move Cost before Drill;
evaluate;
recall;
```

The messages, which appear on the SAS log, show that if there is some way that the wildcatter knows what the cost to drill will be before his decision has to be made, it will yield an expected payoff of $150,000. So, the expected value of perfect information about drilling cost is $150,000 - $140,000 = $10,000.

```
NOTE: The current problem has been successfully saved.

NOTE: Present order of stages:

      Cost(C), Drill(D), Oil_Deposit(C), _ENDST_(E).

NOTE: The currently optimal decision yields 150000.

NOTE: The original problem has been successfully recalled.
NOTE: Present order of stages:

      Drill(D), Cost(C), Oil_Deposit(C), _ENDST_(E).
```

Value of Perfect Control

The oil wildcatter may also want to know what the value of perfect control (VPC) is on the cost of drilling. That is, how much is he willing to pay for getting complete control on the drilling cost? This analysis can be performed with the following statement:

```
vpc Cost;
```

The result is written to the SAS log as

```
NOTE: The currently optimal decision yields 140000.
NOTE: The new optimal decision yields 300000.
NOTE: The value of perfect control of stage Cost
      yields 160000.
```

Oil Wildcatter's Problem with Sounding Test

The wildcatter is impressed with the results of calculating the values of perfect information and perfect control. After comparing those values with the costs of the sounding test and the cost-controlling procedure, he prefers to spend $60,000 on sounding test, which has a potential improvement of $155,000. He is informed that the sounding will disclose whether the terrain below has no structure (which is bad), open structure (which is okay), or closed structure (which is really hopeful). The expert also provides him with the following table, which shows the conditional probabilities.

Table 7.2 Conditional Probabilities of Oil Wildcatter's Problem

	Seismic Outcomes		
State	**No Structure**	**Open Structure**	**Closed Structure**
Dry	0.6	0.3	0.1
Wet	0.3	0.4	0.3
Soaking	0.1	0.4	0.5

To include this additional information into his basic problem, the wildcatter needs to add two stages to his model: a decision stage to represent the decision whether or not to take the sounding test, and one chance stage to represent the uncertain test result. The new STAGEIN= data set is

```
/* -- create the STAGEIN= data set              -- */
data Dtoils2;
format _STNAME_ $12. _STTYPE_ $2.  _OUTCOM_ $14.
    _SUCCES_ $12. _REWARD_ dollar12.0;
input _STNAME_ & _STTYPE_ & _OUTCOM_ &
    _SUCCES_ & _REWARD_ dollar12.0;
datalines;
Drill        D   Drill          Cost                .
.            .   Not_Drill      .                   .
Cost         C   Low            Oil_Deposit         .
.            .   Fair           Oil_Deposit         .
.            .   High           Oil_Deposit         .
Oil_Deposit  C   Dry            .                   .
.            .   Wet            .                   .
.            .   Soaking        .                   .
Sounding     D   Noseismic      Drill               .
.            .   Seismic        Structure     -$60,000
Structure    C   No_Struct      Drill               .
.            .   Open_Struct    Drill               .
.            .   Closed_Struct  Drill               .
;
```

Note that the cost for the seismic soundings is represented as negative reward (of the outcome Seismic) in this data set. The conditional probabilities for stage Structure are added to the PROBIN= data set as follows:

```
   /* -- create PROBIN= data set                     -- */
data Dtoilp2;
   format _EVENT1 $10. _EVENT2 $12. _EVENT3 $14. ;
   input _GIVEN_ $ _EVENT1 $ _PROB1
         _EVENT2 $ _PROB2  _EVENT3 $ _PROB3;
   datalines;
.       Low       0.2 Fair        0.6 High           0.2
.       Dry       0.5 Wet         0.3 Soaking        0.2
Dry     No_Struct 0.6 Open_Struct 0.3 Closed_Struct 0.1
Wet     No_Struct 0.3 Open_Struct 0.4 Closed_Struct 0.3
Soaking No_Struct 0.1 Open_Struct 0.4 Closed_Struct 0.5
;
```

It is not necessary to make any change to the PAYOFFS= data set. To evaluate his new model, the wildcatter invokes PROC DTREE as follows:

```
   /* -- PROC DTREE statements                        -- */
title "Oil Wildcatter's Problem";

proc dtree stagein=Dtoils2
           probin=Dtoilp2
           payoffs=Dtoilu1
           nowarning;

   evaluate;
```

As before, the following messages are written to the SAS log:

```
NOTE: Present order of stages:

      Sounding(D), Structure(C), Drill(D), Cost(C),
      Oil_Deposit(C), _ENDST_(E).

NOTE: The currently optimal decision yields 140000.
```

The following SUMMARY statements produce the optimal decision summary as shown in Figure 7.5 and Figure 7.6:

```
   summary / target=Sounding;
   summary / target=Drill;
```

The optimal strategy for the oil-drilling problem is found to be the following:

- No soundings test should be taken, and always drill. This alternative has an expected payoff of $140,000.
- If the soundings test is conducted, then drill unless the test shows the terrain below has no structure.
- The soundings test is worth $180,100 - $140,000 = $40,100 (this quantity is also called the *value of imperfect information* or the *value of sample information*), but it costs $60,000; therefore, it should not be taken.

Figure 7.5 Summary of the Oil Wildcatter's Problem for SOUNDING

Oil Wildcatter's Problem

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Sounding	Decision
Structure	Chance
Drill	Decision
Cost	Chance
Oil_Deposit	Chance
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Expected Value (MAXEV)
Optimal Decision Yields:	$140,000

Optimal Decision Policy Up to Stage Sounding		
Alternatives or Outcomes	**Cumulative Reward**	**Evaluating Value**
Noseismic	$0	$140,000*
Seismic	$-60,000	$180,100

Figure 7.6 Summary of the Oil Wildcatter's Problem for DRILL

Oil Wildcatter's Problem

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Sounding	Decision
Structure	Chance
Drill	Decision
Cost	Chance
Oil_Deposit	Chance
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Expected Value (MAXEV)
Optimal Decision Yields:	$140,000

Optimal Decision Policy
Up to Stage Drill

Alternatives or Outcomes			Cumulative Reward	Evaluating Value
Noseismic		**Drill**	$0	$140,000*
Noseismic		**Not_Drill**	$0	$0
Seismic	**No_Struct**	**Drill**	$-60,000	$-97,805
Seismic	**No_Struct**	**Not_Drill**	$-60,000	$0*
Seismic	**Open_Struct**	**Drill**	$-60,000	$204,286*
Seismic	**Open_Struct**	**Not_Drill**	$-60,000	$0
Seismic	**Closed_Struct**	**Drill**	$-60,000	$452,500*
Seismic	**Closed_Struct**	**Not_Drill**	$-60,000	$0

Note that the value of sample information also can be obtained by using the following statements:

```
modify Seismic reward 0;
evaluate;
```

The following messages, which appear in the SAS log, show the expected payoff with soundings test is $180,100. Recall that the expected value without test information is $140,000. Again, following the previous calculation, the value of test information is $180,100 - $140,000 = $40,100.

```
NOTE: The reward of outcome Seismic has been changed to 0.

NOTE: The currently optimal decision yields 180100.
```

Now, the wildcatter has the information to make his best decision.

Syntax: DTREE Procedure

The following statements are available in PROC DTREE:

PROC DTREE *options* ;
  EVALUATE / *options* ;
  MODIFY *specifications* ;
  MOVE *specifications* ;
  QUIT ;
  RECALL ;
  RESET *options* ;
  SAVE ;
  SUMMARY / *options* ;
  TREEPLOT / *options* ;
  VARIABLES / *options* ;
  VPC *specifications* ;
  VPI *specifications* ;

The DTREE procedure begins with the PROC DTREE statement and terminates with the QUIT statement. The VARIABLES statement can be used only once, and if it is used, it must appear before any other statements. The EVALUATE, MODIFY, MOVE, RECALL, RESET, SAVE, SUMMARY, TREEPLOT, VPC, and VPI statements can be listed in any order and can be used as many times as desired with one exception: the RECALL statement must be preceded by at least one SAVE statement.

You can also submit any other valid SAS statements, for example, OPTIONS, TITLE, and SAS/GRAPH global statements. In particular, the SAS/GRAPH statements that can be used to enhance the DTREE procedure's output on graphics devices are listed in Table 7.3. Note that the DTREE procedure is not supported with the ActiveX or Java series of devices on the GOPTIONS statement. Refer to *SAS/GRAPH Software: Reference* for more explanation of these statements.

Table 7.3 Statements to Enhance Graphics Output

Statement	Function
FOOTNOTE	Produce footnotes that are displayed on the graphics output
GOPTIONS	Define default values for graphics options
NOTE	Produce text that is displayed on the graphics output
SYMBOL	Create symbol definitions
TITLE	Produce titles that are displayed on the graphics output

Functional Summary

Table 7.4 outlines the options available for the DTREE procedure, classified by function.

Table 7.4 Functional Summary

Description	Statement	Option
Accuracy Control Options		
Specifies the accuracy of numerical computation	DTREE, RESET	TOLERANCE=
Data Set Specifications		
Specifies the Annotate data set	DTREE, RESET, TREEPLOT	ANNOTATE=
Specifies the Image map output data set	DTREE, RESET, TREEPLOT	IMAGEMAP=
Specifies the Payoffs data set	DTREE	PAYOFFS=
Specifies the Probability data set	DTREE	PROBIN=
Specifies the Stage data set	DTREE	STAGEIN=
Error Handling Options		
Automatically rescales the probabilities of an uncertainty if they do not sum to 1	DTREE, RESET	AUTOSCALE
Specifies the error handling behavior	DTREE, RESET	ERRHANDLE=
Disables automatic rescaling of probabilities	DTREE, RESET	NOSCALE
Disables warning messages	DTREE, RESET	NOWARNING
Enables warning messages	DTREE, RESET	WARNING
Evaluation Control Options		
Specifies the criterion used to determine the optimal decision	DTREE, EVALUATE, RESET	CRITERION=
Specifies the risk tolerance	DTREE, EVALUATE, RESET	RT=
Format Control Options		
Specifies the maximum decimal width to format numerical values	DTREE, EVALUATE, RESET, SUMMARY, TREEPLOT	MAXPREC=
Specifies the maximum field width to format numerical values	DTREE, EVALUATE, RESET, SUMMARY, TREEPLOT	MAXWIDTH=
Specifies the maximum field width to format names	DTREE, EVALUATE, RESET, SUMMARY, TREEPLOT	NWIDTH=
Graphics Catalog Options		
Specifies the description field for the catalog entry	DTREE, RESET, TREEPLOT	DESCRIPTION=
Specifies the name of the graphics catalog	DTREE, RESET, TREEPLOT	GOUT=
Specifies the name field for the catalog entry	DTREE, RESET, TREEPLOT	NAME=

Table 7.4 *continued*

Description	Statement	Option
Line-Printer Options		
Specifies the characters for line-printer plot	DTREE, RESET, TREEPLOT	FORMCHAR=
Link Appearance Options		
Specifies the color of LOD[1]	DTREE, RESET, TREEPLOT	CBEST=
Specifies the color of all links except LOD[1]	DTREE, RESET, TREEPLOT	CLINK=
Defines the symbol for all links except LOD[1] and LCP[2]	DTREE, RESET, TREEPLOT	LINKA=
Defines the symbol for LOD[1]	DTREE, RESET, TREEPLOT	LINKB=
Defines the symbol for LCP[2]	DTREE, RESET, TREEPLOT	LINKC=
Specifies the line type of all links except LOD[1] and LCP[2]	DTREE, RESET, TREEPLOT	LSTYLE=
Specifies the line type of LOD[1]	DTREE, RESET, TREEPLOT	LSTYLEB=
Specifies the line type of LCP[2]	DTREE, RESET, TREEPLOT	LSTYLEC=
Specifies the line thickness of all links except LOD[1]	DTREE, RESET, TREEPLOT	LWIDTH=
Specifies the line thickness of LOD[1]	DTREE, RESET, TREEPLOT	LWIDTHB=
Node Appearance Options		
Specifies the color of chance nodes	DTREE, RESET, TREEPLOT	CSYMBOLC=
Specifies the color of decision nodes	DTREE, RESET, TREEPLOT	CSYMBOLD=
Specifies the color of end nodes	DTREE, RESET, TREEPLOT	CSYMBOLE=
Specifies the height of symbols for all nodes	DTREE, RESET, TREEPLOT	HSYMBOL=
Specifies the symbol definition for chance nodes	DTREE, RESET, TREEPLOT	SYMBOLC=
Specifies the symbol definition for decision nodes	DTREE, RESET, TREEPLOT	SYMBOLD=
Specifies the symbol definition for end nodes	DTREE, RESET, TREEPLOT	SYMBOLE=
Specifies the symbol used for chance nodes	DTREE, RESET, TREEPLOT	VSYMBOLC=
Specifies the symbol used for decision nodes	DTREE, RESET, TREEPLOT	VSYMBOLD=
Specifies the symbol used for end nodes	DTREE, RESET, TREEPLOT	VSYMBOLE=
Output Control Options		
Suppresses display of the optimal decision summary	DTREE, EVALUATE, RESET	NOSUMMARY
Displays the optimal decision summary	DTREE, EVALUATE, RESET	SUMMARY

Table 7.4 *continued*

Description	Statement	Option
Specifies the decision stage up to which the optimal decision summary is displayed	DTREE, EVALUATE, RESET, SUMMARY	TARGET=
Plot Control Options		
Draws diagram on one page in graphics mode	DTREE, RESET, TREEPLOT	COMPRESS
Displays information on the decision tree diagram	DTREE, RESET, TREEPLOT	DISPLAY=
Processes the Annotate data set	DTREE, RESET, TREEPLOT	DOANNOTATE
Invokes graphics version	DTREE, RESET, TREEPLOT	GRAPHICS
Displays labels	DTREE, RESET, TREEPLOT	LABEL
Displays legend	DTREE, RESET, TREEPLOT	LEGEND
Invokes line-printer version	DTREE, RESET, TREEPLOT	LINEPRINTER
Suppresses processing of the Annotate data set	DTREE, RESET, TREEPLOT	NOANNOTATE
Draws diagram across multiple pages	DTREE, RESET, TREEPLOT	NOCOMPRESS
Suppresses displaying label	DTREE, RESET, TREEPLOT	NOLABEL
Suppresses displaying legend	DTREE, RESET, TREEPLOT	NOLEGEND
Suppresses displaying page number	DTREE, RESET, TREEPLOT	NOPAGENUM
Uses rectangular corners for turns in the links	DTREE, RESET, TREEPLOT	NORC
Displays page number at upper right corner	DTREE, RESET, TREEPLOT	PAGENUM
Uses rounded corners for turns in the links	DTREE, RESET, TREEPLOT	RC
Specifies the vertical space between two end nodes	DTREE, RESET, TREEPLOT	YBETWEEN=
Text Appearance Options		
Specifies the text color	DTREE, RESET, TREEPLOT	CTEXT=
Specifies the text font	DTREE, RESET, TREEPLOT	FTEXT=
Specifies the text height	DTREE, RESET, TREEPLOT	HTEXT=
Variables in PAYOFFS= Data Set		
Specifies the action outcome names	VARIABLES	ACTION=
Specifies the state outcome names	VARIABLES	STATE=
Specifies the payoffs	VARIABLES	VALUE=
Variables in PROBIN= Data Set		
Specifies the event outcome names	VARIABLES	EVENT=
Specifies the given outcome names	VARIABLES	GIVEN=
Specifies the (conditional) probabilities	VARIABLES	PROB=

Table 7.4 *continued*

Description	Statement	Option
Variables in STAGEIN= Data Set		
Specifies the outcome names	VARIABLES	OUTCOME=
Specifies the rewards	VARIABLES	REWARD=
Specifies the stage name	VARIABLES	STAGE=
Specifies the successor names	VARIABLES	SUCCESSOR=
Specifies the type of stage	VARIABLES	TYPE=
Specifies the web reference variable	VARIABLES	WEB=

[1]LOD denotes links that indicate optimal decisions.

[2]LCP denotes links that continue on subsequent pages.

PROC DTREE Statement

PROC DTREE *options* **;**

The options that can appear in the PROC DTREE statement are listed in the following section. The options specified in the PROC DTREE statement remain in effect for all statements until the end of processing or until they are changed by a RESET statement. These options are classified under appropriate headings: first, all options that are valid for all modes of the procedure are listed followed by the options classified according to the mode (line-printer or graphics) of invocation of the procedure.

General Options

AUTOSCALE | NOSCALE
specifies whether the procedure should rescale the probabilities of events for a given chance stage if the total probability of this stage is not equal to 1. The default is NOSCALE.

CRITERION=*i*
indicates the decision criterion to be used for determining the optimal decision and the certain equivalent for replacing uncertainties. The following table shows all valid values of *i* and their corresponding decision criteria and certain equivalents.

Table 7.5 Values for the CRITERION= Option

i	Criterion	Certain Equivalent
MAXEV	Maximize	Expected value
MINEV	Minimize	Expected value
MAXMLV	Maximize	Value with largest probability
MINMLV	Minimize	Value with largest probability
MAXCE	Maximize	Certain equivalent value of expected utility
MINCE	Minimize	Certain equivalent value of expected utility

The default value is MAXEV. The last two criteria are used when your utility curve can be fit by an exponential function. See the section "Evaluation" on page 425 for more information on the exponential utility function.

DISPLAY=*(information-list)*
specifies information that should be displayed on each link of the decision tree diagram. Table 7.6 lists the valid keywords and corresponding information.

Table 7.6 Information on Decision Tree and Keywords

Keyword	Information
ALL	All information listed in this table
CR	Cumulative rewards of outcomes on the path that leads to the successor of the link
EV	Evaluating value that can be expected from the successor of the link
LINK	Outcome name represented by the link
P	Probability of the outcome represented by the link
R	Instant reward of the outcome represented by the link
STAGE	Stage name of the successor of the link

The default value is (LINK P EV R CR).

Note that the probability information displays on links that represent chance outcomes only. In addition, the PROBIN= option must be specified. The expected values display only if the decision tree has been evaluated. The reward information displays on a link only if the instant reward of the outcome represented by the link is nonzero. The cumulative rewards do not display if the cumulative rewards of links are all zero.

ERRHANDLE=DRAIN | QUIT
specifies whether the procedure should stop processing the current statement and wait for next statement or quit PROC DTREE when an error has been detected by the procedure. The default value is DRAIN.

GRAPHICS
creates plots for a graphics device. To specify this option, you need to have SAS/GRAPH software licensed at your site. This is the default.

LABEL | NOLABEL
specifies whether the labels for information displayed on the decision tree diagram should be displayed. If the NOLABEL option is not specified, the procedure uses the following symbols to label all the information that is displayed on each link.

Table 7.7 Labels and Their Corresponding Information

Label	Information
cr=	The cumulative rewards of outcomes on the path that lead to the successor of the link
EV=	The value that can be expected from the successor of the link
p=	The probability of the outcome represented by the link
r=	The instant reward of the outcome

The default is LABEL.

LINEPRINTER

LP

creates plots of line-printer quality. If you do not specify this option, graphics plots are produced.

MAXPREC=*d*

specifies the maximum decimal width (the precision) in which to format numerical values using *w.d* format. This option is used in displaying the decision tree diagrams and the summaries. The value for this option must be no greater than 9; the default value is 3.

MAXWIDTH=*mw*

specifies the maximum field width in which to format numerical values (probabilities, rewards, cumulative rewards and evaluating values) using $w.d$ format. This option is used in displaying the decision tree diagrams and the summaries. The value for this option must be no greater than 16 and must be at least 5 plus the value of the MAXPREC= option. The default value is 10.

NWIDTH=*nw*

specifies the maximum field width in which to format outcome names when displaying the decision tree diagrams. The value for this option must be no greater than 40; the default value is 32.

PAYOFFS=*SAS-data-set*

names the SAS data set that contains the evaluating values (payoffs, losses, utilities, and so on) for each state and action combination. The use of PAYOFFS= is optional in the PROC DTREE statement. If the PAYOFFS= option is not used, PROC DTREE assumes that all evaluating values at the end nodes of the decision tree are 0.

PROBIN=*SAS-data-set*

names the SAS data set that contains the (conditional) probability specifications of outcomes. The PROBIN= SAS data set is required if the evaluation of the decision tree is desired.

RT=*r*

specifies the value of the risk tolerance. The RT= option is used only when CRITERION=MAXCE or CRITERION=MINCE is specified. If the RT= option is not specified, and CRITERION=MAXCE or CRITERION=MINCE is specified, PROC DTREE changes the value of the CRITERION= option to MAXEV or MINEV (which would mean straight-line utility function and imply infinite risk tolerance).

STAGEIN=*SAS-data-set*

names the SAS data set that contains the stage names, stage types, names of outcomes, and their rewards and successors for each stage. If the STAGEIN= option is not specified, PROC DTREE uses the most recently created SAS data set.

SUMMARY | NOSUMMARY

specifies whether an optimal decision summary should be displayed each time the decision tree is evaluated. The decision summary lists all paths through the tree that lead to the target stage as well as the cumulative rewards and the evaluating values of all alternatives for that path. The alternative with optimal evaluating value for each path is marked with an asterisk (*). The default is NOSUMMARY.

TARGET=*stage*

specifies the decision stage up to which the optimal decision policy table is displayed. The TARGET= option is used only in conjunction with the SUMMARY option. The stage specified must be a decision stage. If the TARGET= option is not specified, the procedure displays an optimal decision policy table for each decision stage.

TOLERANCE=*d*

specifies either a positive number close to 0 or greater than 1. PROC DTREE treats all numbers within e of 0 as 0, where

$$e = \begin{cases} d & \text{if } d < 1 \\ d \times \epsilon & \text{otherwise} \end{cases}$$

and ϵ is the *machine epsilon*. The default value is 1,000.

WARNING | NOWARNING

specifies whether the procedure should display a warning message when

- the payoff for an outcome is not assigned in the PAYOFFS= data set
- probabilities of events for a given chance stage have been automatically scaled by PROC DTREE because the total probability of the chance stage does not equal 1

The default is WARNING.

YBETWEEN=*ybetween < units >*

specifies the vertical distance between two successive end nodes. If the GRAPHICS option is specified, the valid values for the optional *units* are listed in Table 7.8.

Table 7.8 Valid Values for the Units of the YBETWEEN= Option

Unit	Description
CELL	Character cells
CM	Centimeters
INCH	Inches
PCT	Percentage of the graphics output area
SPACE	Height of the box surrounding the node, its predecessor link, and all text information

The value of the YBETWEEN= option must be greater than or equal to 0. Note that if the COMPRESS option is specified, the actual distance between two successive end nodes is scaled by PROC DTREE and may not be the same as the YBETWEEN= specification.

If the LINEPRINTER option is specified, the optional *units* value can be CELL or SPACE. The value of the YBETWEEN= option must be a nonnegative integer.

If you do not specify units, a unit specification is determined in the following order:

- the GUNIT= option in a GOPTIONS statement, if the GRAPHICS option is specified
- the default unit, CELL

The default value of YBETWEEN= option is 0.

Graphics Options

The following options are specifically for the purpose of producing a high-resolution quality decision tree diagram.

ANNOTATE=*SAS-data-set*

ANNO=*SAS-data-set*

specifies an input data set that contains appropriate Annotate variables. The ANNOTATE= option enables you to add features (for example, customized legend) to plots produced on graphics devices. For additional information, refer to the chapter on the annotate data set in *SAS/GRAPH Software: Reference*.

CBEST=*color*

CB=*color*

specifies the color for all links in the decision tree diagram that represent optimal decisions. If you do not specify the CBEST= option, the color specification is determined in the following order:

- the CI= option in the *j*th generated SYMBOL definition, if the option LINKB=*j* is specified
- the ContrastColor attribute of the GraphData2 element of the current ODS style template (if the GSTYLE system option is active)
- the second color in the colors list

CLINK=*color*

CL=*color*

specifies the color for all links in the decision tree diagram except those that represent optimal decisions. If the CLINK= option is not specified, the color specification is determined in the following order:

- the CI= option in the *i*th generated SYMBOL definition, if the option LINKA=*i* is specified
- the ContrastColor attribute of the GraphData3 element of the current ODS style template (if the GSTYLE system option is active)
- the third color in the colors list

COMPRESS | NOCOMPRESS

CP | NOCP

specifies whether the decision tree diagram should be drawn on one physical page. If the COMPRESS option is specified, PROC DTREE determines the scale so that the diagram is compressed, if necessary, to fit on one physical page. Otherwise, the procedure draws the diagram across multiple pages if necessary. The default is NOCOMPRESS.

CSYMBOLC=*color*

CC=*color*

specifies the color of the symbol used to draw all chance nodes in the decision tree diagram. If the CSYMBOLC= option is not specified, the color specification is determined in the following order:

- the CV= option in the *m*th generated SYMBOL definition, if the option SYMBOLC=*m* is specified
- the CSYMBOL= option in a GOPTIONS statement

- the ContrastColor attribute of the GraphData1 element of the current ODS style template (if the GSTYLE system option is active)
- the fifth color in the colors list

CSYMBOLD=*color*

CD=*color*

specifies the color of the symbol used to draw all decision nodes in the decision tree diagram. If the CSYMBOLD= option is not specified, the color specification is determined in the following order:

- the CV= option in the *d*th generated SYMBOL definition, if the option SYMBOLD=*d* is specified
- the CSYMBOL= option in a GOPTIONS statement
- the ContrastColor attribute of the GraphData5 element of the current ODS style template (if the GSTYLE system option is active)
- the fourth color in the colors list

CSYMBOLE=*color*

CE=*color*

specifies the color of the symbol used to draw all end nodes in the decision tree diagram. If the CSYMBOLE= option is not specified, the color specification is determined in the following order:

- the CV= option in the *n*th generated SYMBOL definition, if the option SYMBOLE=*n* is specified
- the CSYMBOL= option in a GOPTIONS statement
- the ContrastColor attribute of the GraphData8 element of the current ODS style template (if the GSTYLE system option is active)
- the sixth color in the colors list

CTEXT=*color*

CT=*color*

specifies the color to be used for all text that appears on plots except on TITLE and FOOTNOTE lines. If the CTEXT= option is not specified, the color specification is determined in the following order:

- the CTEXT= option in a GOPTIONS statement
- the Color attribute of the GraphDataText element of the current ODS style template (if the GSTYLE system option is active)
- the first color in the colors list

DESCRIPTION=*'string'*

DES=*'string'*

specifies a descriptive string, up to 40 characters long, that appears in the description field of the master menu of PROC GREPLAY. If the DESCRIPTION= option is omitted, the description field contains a description assigned by PROC DTREE.

DOANNOTATE | NOANNOTATE

DOANNO | NOANNO

specifies whether the Annotate data set should be processed. If the NOANNOTATE option is specified, the procedure does not process the Annotate data set even though the ANNOTATE= option is specified. The default is DOANNOTATE.

FTEXT=*name*

FONT=*name*

specifies the font to be used for text on plots. If you do not use this option, the font specification is determined in the following order:

- the FTEXT= option in a GOPTIONS statement
- the Font attribute of the GraphDataText element of the current ODS style template (if the GSTYLE system option is active)
- the hardware font for your graphics output device

Refer to the chapter on SAS/GRAPH fonts in *SAS/GRAPH Software: Reference* for details about SAS/GRAPH fonts.

GOUT=*SAS-catalog*

specifies the name of the graphics catalog used to save the output produced by PROC DTREE for later replay. For additional information, refer to the chapter on graphics output in *SAS/GRAPH Software: Reference*.

HSYMBOL=*h*

HS=*h*

specifies that the height of symbols for all nodes in the decision tree diagram is *h* times the heights of symbols assigned by SAS/GRAPH software. You can specify the heights of decision nodes, chance nodes, and end nodes by using the HEIGHT= options in the corresponding SYMBOL statements. For example, if you specify the options HSYMBOL=*2* and SYMBOLD=*1* in the PROC DTREE statement and defined SYMBOL*1* as

```
symbol1 height=4 pct;
```

then all decision nodes in the decision tree diagram are sized at $2 \times 4 = 8\%$ of the graphics output area. The default value is 1.

HTEXT=*h*

HT=*h*

specifies that the height for all text in plots (except that in TITLE and FOOTNOTE statements) be *h* times the height of the characters assigned by SAS/GRAPH software. You can also specify character height by using the HTEXT= option in a GOPTIONS statement.

For example, if you specify the option HTEXT=0.6 in the PROC DTREE statement and also specified a GOPTIONS statement as follows

```
goptions htext=2 in;
```

then the size of all text is $0.6 \times 2 = 1.2$ inches. For more explanation of the GOPTIONS statement, refer to the chapter on the GOPTIONS statement in *SAS/GRAPH Software: Reference*. The default value is 1.

IMAGEMAP=*SAS-data-set*

names the SAS data set that receives a description of the areas of a graph and a link for each area. This information is for the construction of HTML image maps. You use a SAS DATA step to process the output file and generate your own HTML files. The graph areas correspond to the link information that comes from the WEB= variable in the STAGEIN= data set. This gives you complete control over the appearance and structure of your HTML pages.

LEGEND | NOLEGEND

LG | NOLG

specifies whether the default legend should be displayed. If the NOLEGEND is not specified, the procedure displays a legend at the end of each page of the decision tree diagram. The default is LEGEND.

LINKA=*i*

If the LINKA=*i* option is specified, then PROC DTREE uses the color specified with the CI= option, the type specified with the LINE= option, and the thickness specified with the WIDTH= option in the *i*th generated SYMBOL definition to draw all links in the decision tree diagram, except those that indicate optimal decisions and those that are continued on subsequent pages. There is no default value for this option. The color, type, and thickness specifications may be overridden by the specifications of the CLINK=, LSTYLE=, and LWIDTH= options in the PROC DTREE statement.

Note that if you specify the LINKA=*i* option, PROC DTREE uses the specifications in the *i*th *generated* SYMBOL *definition* and not the specifications in the SYMBOL*i* statement. Refer to *SAS/GRAPH Software: Reference* for the details about creating, canceling, reviewing, and altering SYMBOL definitions.

LINKB=*j*

If the LINKB=*j* option is specified, then PROC DTREE uses the color specified with the CI= option, the type specified with the LINE= option, and the thickness specified with the WIDTH= option in the *j*th generated SYMBOL definition to draw all links that represent optimal decisions. There is no default value for this option. The color, type, and thickness specifications may be overridden by the specifications of the CBEST=, LSTYLEB=, and LWIDTHB= options in the PROC DTREE statement.

Note that if you specify the LINKB=*j* option, PROC DTREE uses the specifications in the *j*th *generated* SYMBOL *definition* and not the specifications in the SYMBOL*j* statement. Refer to *SAS/GRAPH Software: Reference* for the details about creating, canceling, reviewing, and altering SYMBOL definitions.

LINKC=*k*

If the LINKC=*k* option is specified, then PROC DTREE uses the type specified with the LINE= option in the *k*th generated SYMBOL definition to draw all links in the decision tree diagram that are continued on subsequent pages. There is no default value for this option. The color and thickness for links continued on another page indicate whether the link represents an optimal decision or not. The type specification may be overridden by the specification of the LSTYLEC= option in the PROC DTREE statement.

Note that if you specify the LINKC=*k* option, PROC DTREE uses the specifications in the *k*th *generated* SYMBOL *definition* and not the specifications in the SYMBOL*k* statement. Refer to *SAS/GRAPH Software: Reference* for the details about creating, canceling, reviewing, and altering SYMBOL definitions.

LSTYLE=*l*
L=*l*

specifies the line type (style) used for drawing all links in the decision tree diagram, except those that represent the optimal decisions and those that are continued on subsequent pages. Valid values for *l* are 1 through 46. If the LSTYLE= option is not specified, the type specification is determined in the following order:

- the LINE= option in the *i*th generated SYMBOL definition, if the option LINKA=*i* is specified
- the default value, 1 (solid line)

LSTYLEB=*l2*
LB=*l2*

specifies the line type (style) used for drawing the links in the decision tree diagram that represent optimal decisions. Valid values for *l2* are 1 through 46. If the LSTYLEB= option is not specified, the type specification is determined in the following order:

- the LINE= option in the *j*th generated SYMBOL definition, if the option LINKB=*j* is specified
- the default value, 1 (solid line)

LSTYLEC=*l3*
LC=*l3*

specifies the line type (style) used for drawing the links in the decision tree diagram that are continued on the next subsequent pages. Valid values for *l3* are 1 through 46.

If the LSTYLEC= option is not specified, the type specification is determined in the following order:

- the LINE= option in the *k*th generated SYMBOL definition, if the option LINKC=*k* is specified
- the default value, 2 (dot line)

LWIDTH=*w*
LTHICK=*w*

specifies the line thickness (width) used to draw all links in the decision tree diagram except those that represent the optimal decisions.

If the LWIDTH= option is not specified, the thickness specification is determined in the following order:

- the WIDTH= option in the *i*th generated SYMBOL definition, if the option LINKA=*i* is specified
- the default value, 1

LWIDTHB=*w2*
LTHICKB=*w2*

specifies the line thickness (width) used to draw the links in the decision tree diagram that represent optimal decisions. If the LWIDTHB= option is not specified, the thickness specification is determined in the following order:

- the WIDTH= option in the *j*th generated SYMBOL definition, if the option LINKB=*j* is specified
- 2 times the thickness for links that represent regular outcomes

NAME=*'string'*

specifies a descriptive string, up to 8 characters long, that appears in the name field of the master menu of PROC GREPLAY. The default is 'DTREE '.

PAGENUM | NOPAGENUM

PAGENUMBER | NOPAGENUMBER

specifies whether the page numbers should be displayed in the top right corner of each page of a multipage decision tree diagram. If the NOPAGENUM is not specified, the pages are ordered from top to bottom, left to right.

The default is PAGENUM.

RC | NORC

specifies whether the links in the decision tree diagram should be drawn with rounded corners or with rectangular corners. The default is RC.

SYMBOLC=*m*

SYMBC=*m*

If the SYMBOLC= option is specified, then PROC DTREE uses the color specified with the CV= option, the character specified with the VALUE= option, the font specified with the FONT= option, and the height specified with the HEIGHT= option in the *m*th generated SYMBOL definition to draw all chance nodes in the decision tree diagram. There is no default value for this option. The color and the symbol specifications may be overridden by the specification of the CSYMBOLC= and VSYMBOLC= options in the PROC DTREE statement. The height of the symbol can be changed by the HSYMBOL= option in the PROC DTREE statement.

Note that if you specify the SYMBOLC=*m* option, PROC DTREE uses the specifications in the *m*th generated SYMBOL definition and not the specifications in the SYMBOL*m* statement. Refer to *SAS/GRAPH Software: Reference* for the details about creating, canceling, reviewing, and altering SYMBOL definitions.

SYMBOLD=*d*

SYMBD=*d*

If the SYMBOLD= option is specified, then PROC DTREE uses the color specified with the CV= option, the character specified with the VALUE= option, the font specified with the FONT= option, and the height specified with the HEIGHT= option in the *d*th generated SYMBOL definition to draw all decision nodes in the decision tree diagram. There is no default value for this option. The color and the symbol specifications may be overridden by the specification of the CSYMBOLD= and VSYMBOLD= options in the PROC DTREE statement. The height of the characters can be changed by the HSYMBOL= option in the PROC DTREE statement.

Note that if you specify the SYMBOLD=*d* option, PROC DTREE uses the specifications in the *d*th generated SYMBOL definition and not the specifications in the SYMBOL*d* statement. Refer to *SAS/GRAPH Software: Reference* for the details about creating, canceling, reviewing, and altering SYMBOL definitions.

SYMBOLE=*n*

SYMBE=*n*

If the SYMBOLE= option is specified, then PROC DTREE uses the color specified with the CV= option, the character specified with the VALUE= option, the font specified with the FONT= option, and the height specified with the HEIGHT= option in the *n*th generated SYMBOL definition to draw all end nodes in the decision tree diagram. There is no default value for this option. The color and the symbol specifications may be overridden by the specification of the CSYMBOLE= and VSYMBOLE= options specified in the PROC DTREE statement. The height of the characters can be changed by the HSYMBOL= option in the PROC DTREE statement.

Note that if you specify the SYMBOLE=*n* option, PROC DTREE uses the specifications in the *n*th generated SYMBOL definition and not the specifications in the SYMBOL*n* statement. Refer to *SAS/GRAPH Software: Reference* for the details about creating, canceling, reviewing, and altering SYMBOL definitions.

VSYMBOLC=*symbolc-name*

VC=*symbolc-name*

specifies that the symbol *symbolc-name* from the special symbol table be used to draw all chance nodes in the decision tree diagram. If you do not specify this option, the symbol used is determined in the following order:

- the options VALUE= and FONT= specifications in the *m*th generated SYMBOL definition, if the option SYMBOLC=*m* is specified
- the symbol CIRCLE in the special symbol table

VSYMBOLD=*symbold-name*

VD=*symbold-name*

specifies that the symbol *symbold-name* from the special symbol table be used to draw all decision nodes in the decision tree diagram. If you do not specify this option, the symbol used is determined in the following order:

- the options VALUE= and FONT= specifications in the *d*th generated SYMBOL definition, if the option SYMBOLD=*d* is specified
- the symbol SQUARE in the special symbol table

VSYMBOLE=*symbole-name*

VE=*symbole-name*

specifies that the symbol *symbole-name* from the special symbol table be used to draw all end nodes in the decision tree diagram. If you do not specify this option, the symbol used is determined in the following order:

- the options VALUE= and FONT= specifications in the *n*th generated SYMBOL definition, if the option SYMBOLE=*n* is specified
- the symbol DOT in the special symbol table

Line-Printer Options

The following options are specifically for the purpose of producing line-printer quality decision tree diagram.

FORMCHAR< *(syni-list)* >**=** *'formchar-string'*

defines characters to be used for features on line-printer plots. The *syni-list* is a list of numbers ranging from 1 to 13. The list identifies which features are controlled with the string characters. The *formchar-string* gives characters for features in *syni-list*. Any character or hexadecimal string can be used. By default, *syni-list* is omitted, and the FORMCHAR= option gives a string for all 13 features. The features associated with values of *syni* are listed in Table 7.9. Note that characters 4, 6, 7, 10, and 12 are not used in drawing a decision tree diagram.

Table 7.9 Features Associated with the FORMCHAR= Option

Syni	Description of Character	Feature
1	Vertical bar	Vertical link
2	Horizontal bar	Horizontal link
3	Box character (upper left)	Vertical up to horizontal turn
5	Box character (upper right)	Horizontal and down vertical joint
8	Box character (middle right)	Horizontal to split joint
9	Box character (lower left)	Vertical down to horizontal turn
11	Box character (lower right)	Horizontal and up vertical joint
13	Horizontal thick	Horizontal link that represents optimal decision

As an example, the decision tree diagram in Figure 7.7 is produced by the following statement:

```
title "Decision Tree Showing the Effects of FORMCHAR";

data Dtoils4;
   format _STNAME_ $12. _STTYPE_  $2. _OUTCOM_ $10.
         _SUCCES_ $12.;
   input _STNAME_ $ _STTYPE_ $ _OUTCOM_ $ _SUCCES_ $;
   datalines;
Drill        D   Drill       Cost
.            .   Not_Drill   .
Cost         C   Low         Oil_Deposit
.            .   High        Oil_Deposit
Oil_Deposit  C   Dry         .
.            .   Wet         .
;

proc dtree stagein=Dtoils4
           nowarning
           ;
   treeplot / formchar(1 2 3 5 8 9 11 13)='|-/*<\+='
              lineprinter display=(LINK);

quit;
```

Figure 7.7 Decision Tree Showing the Effects of FORMCHAR

Decision Tree Showing the Effects of FORMCHAR

The DTREE Procedure

Line-printer Plot

```
                            Low          Dry
                         /----------C-*----------E
               Drill     |            |   Wet
            /----------C-<            \----------E
            |            | High          Dry
----------D-<            \----------C-*----------E
            |                         |   Wet
            |                         \----------E
            |Not_Drill
            \----------E
```

By default, the form character list specified with the SAS system option FORMCHAR= is used; otherwise, the default is `'|----|+|---+='`. Refer to the chapter on the Calendar Procedure in the *SAS Procedures Guide* for more information.

EVALUATE Statement

EVALUATE */ options* **;**

The EVALUATE statement causes PROC DTREE to evaluate the decision tree and calculate the optimal decisions. If the SUMMARY option is specified a decision summary is displayed. Otherwise, the current optimal value is displayed on the SAS log.

The following options, which can appear in the PROC DTREE statement, can also be specified in the EVALUATE statement:

CRITERION=*i*	MAXPREC=*d*	MAXWIDTH=*mw*
NOSUMMARY	NWIDTH=*nw*	RT=*r*
SUMMARY	TARGET=*stage*	

The MAXPREC=, MAXWIDTH=, and NWIDTH=, options are valid only in conjunction with the SUMMARY option. The RT= option is valid only in conjunction with the CRITERION=MAXCE or CRITERION=MINCE specification. The options specified in this statement are only in effect for this statement.

MODIFY Statement

MODIFY *outcome-name REWARD new-value* **;**

MODIFY *stage-name TYPE* **;**

The MODIFY statement is used to change either the type of a stage or the reward from an outcome. If MODIFY *outcome-name* REWARD *new-value* is given where the *outcome-name* is an outcome specified in the STAGEIN= data set, and *new-value* is a numeric value, then the reward of the outcome named *outcome-name* is changed to *new-value*.

If MODIFY *stage-name* TYPE is given where *stage-name* is a stage name specified in the STAGEIN= data set, then the type of the stage named *stage-name* is changed to **'DECISION'** if its current type is **'CHANCE'** and is changed to **'CHANCE'** if its current type is **'DECISION'**. You cannot change the type of an **'END'** stage. The change of the type of a stage from **'CHANCE'** to **'DECISION'** can help the decision-maker learn how much improvement can be expected if he or she could pick which of the future (or unknown) outcomes would occur. However, if you want to change the type of a stage from **'DECISION'** to **'CHANCE'**, the procedure is not able to determine the probabilities for its outcomes unless you specify them in the PROBIN= data set.

MOVE Statement

MOVE *stage1 (BEFORE | AFTER) stage2* **;**

The MOVE statement is used to change the order of the stages. After all data in input data sets have been read, PROC DTREE determines the order (from left to right) of all stages specified in the STAGEIN= data set and display the order in the SAS log. The ordering is determined based on the rule that if stage **A** is the successor of an outcome of stage **B**, then stage **A** should occur to the right of stage **B**. The MOVE statement can be used to change the order. If the keyword BEFORE is used, *stage1* becomes the new successor for all immediate predecessors of *stage2*, and *stage2* becomes the new successor for all outcomes of *stage1*. An outcome is said to be an *immediate predecessor* of a stage if the stage is the successor of that outcome. Similarly, if the keyword AFTER is used, the old leftmost (in previous order) successor of outcomes for *stage2* becomes the new successor for all outcomes of *stage1* and the new successor of all outcomes of *stage2* is *stage1*.

There are two limitations: the END stage cannot be moved, and no stage can be moved after the END stage. In practice, any stage after the END stage is useless.

QUIT Statement

QUIT ;

The QUIT statement tells the DTREE procedure to terminate processing. This statement has no options.

RECALL Statement

RECALL ;

This statement tells PROC DTREE to recall the decision model that was saved previously with a SAVE statement. The RECALL statement has no options.

RESET Statement

RESET *options* **;**

The RESET statement is used to change options after the procedure has started. All of the options that can be set in the PROC DTREE statement can also be reset with this statement, except for the STAGEIN=, the PROBIN=, and the PAYOFFS= data set options.

SAVE Statement

SAVE ;

The SAVE statement saves the current model (attributes of stages and outcomes, the ordering of stages, and so on) to a scratch space from which you can call it back later. It is a good idea to save your decision model before you specify any MOVE or MODIFY statements. Then you can get back to your original model easily after a series ofstatements that change the decision model. The SAVE statement has no options.

SUMMARY Statement

SUMMARY */ options* **;**

Unlike the SUMMARY option on the PROC DTREE statement or the EVALUATE statement, which specifies that PROC DTREE display a decision summary when the decision tree is evaluated, the SUMMARY statement causes the procedure to display the summary immediately. If the decision tree has not been evaluated yet, or if it has been changed (by the MOVE, MODIFY, or RECALL statement) since last evaluated, the procedure evaluates or re-evaluates the decision tree before the summary is displayed.

The following options that can appear in the PROC DTREE statement can also be specified in this statement:

MAXPREC=*d*	MAXWIDTH=*mw*
NWIDTH=*nw*	TARGET=*stage*

The options specified in this statement are in effect only for this statement.

TREEPLOT Statement

TREEPLOT */ options* **;**

The TREEPLOT statement plots the current decision tree (a diagram of the decision problem). Each path in the decision tree represents a possible scenario of the problem. In addition to the nodes and links on the decision tree, the information for each link that can be displayed on the diagram is listed in Table 7.10.

Table 7.10 Information on Decision Tree Diagram

Information	**Labeled by**
Stage name for the successor of the link	NL[3]
Outcome name for the link	NL[3]
Probability of the outcome	p=
Value can be expected from the successor	EV=
Instant reward of the outcome	r=
Cumulative rewards of outcomes on the path that leads to the successor	cr=

[3]NL denotes that this information is not labeled.

If necessary, the outcome names and the stage names are displayed above the link, and other information (if there is any) is displayed below the link. The DISPLAY= option can be used to control which information should be included in the diagram. The NOLABEL can be used to suppress the displaying of the labels.

If the LINEPRINTER option is used, the decision nodes, chance nodes, and the end nodes are represented by the characters '**D**', '**C**', and '**E**', respectively. The links are displayed using the specifications of the FORMCHAR= option. See the section "PROC DTREE Statement" on page 401 for more details. In graphics mode, the control of the appearances of nodes and links is more complex. See the section "Displaying the Decision Tree" on page 429 for more information.

The following options that can appear in the PROC DTREE statement can also be specified in the TREEPLOT statement:

DISPLAY=*(information-list)*	GRAPHICS	LABEL
LINEPRINTER	MAXPREC=*d*	MAXWIDTH=*mw*
NOLABEL	NWIDTH=*nw*	YBETWEEN=*ybetween <units>*

The following line-printer options that can appear in the PROC DTREE statement can also be specified in the TREEPLOT statement if the LINEPRINTER option is specified:

FORMCHAR*<(syni-list)>='formchar-string'*

Moreover, the following graphics options that can appear in the PROC DTREE statement can also be specified in the TREEPLOT statement if the GRAPHICS option is specified:

ANNOTATE=*SAS-data-set*	CBEST=*color*	CLINK=*color*
COMPRESS	CSYMBOLC=*color*	CSYMBOLD=*color*
CSYMBOLE=*color*	CTEXT=*color*	DESCRIPTION=*'string'*
DOANNOTATE	FTEXT=*name*	GOUT=*SAS-catalog*
HSYMBOL=*h*	HTEXT=*h*	IMAGEMAP=*SAS-data-set*
LEGEND	LINKA=*i*	LINKB=*j*
LINKC=*k*	LSTYLE=*l*	LSTYLEB=*l2*
LSTYLEC=*l3*	LWIDTH=*w2*	LWIDTHB=*w2*
NAME=*'string'*	NOANNOTATE	NOCOMPRESS
NOLEGEND	NOPAGENUM	NORC
PAGENUM	RC	SYMBOLC=*m*
SYMBOLD=*d*	SYMBOLE=*n*	VSYMBOLC=*symbolc-name*
VSYMBOLD=*symbold-name*	VSYMBOLE=*symbole-name*	

The options specified in this statement are in effect only for this statement, and they may override the options specified in the PROC DTREE statement.

VARIABLES Statement

VARIABLES */ options* **;**

The VARIABLES statement specifies the variable lists in the input data sets. This statement is optional but if it is used, it must appear immediately after the PROC DTREE statement. The options that can appear in the

VARIABLES statement are divided into groups according to the data set in which they occur. Table 7.11 lists all the variables or variable lists associated with each input data set and their types. It also lists the default variables if they are not specified in this statement.

Table 7.11 Input Data Sets and Their Associated Variables

Data Set	Variable	Type[4]	Interpretation	Default
STAGEIN=	OUTCOME=	C/N	Outcome names	Variables with prefix _OUT
	REWARD=	N	Instant reward	Variables with prefix _REW
	STAGE=	C/N	Stage name	_STNAME_
	SUCCESSOR=	as STAGE=	Immediate successors	Variables with prefix _SUCC
	TYPE=	C/N	Stage type	_STTYPE_
	WEB=	C	HTML page for the stage	
PROBIN=	EVENT=	as OUTCOME=	Event names	Variables with prefix _EVEN
	GIVEN=	as OUTCOME=	Names of given outcomes	Variables with prefix _GIVE
	PROB=	N	Conditional probabilities	Variables with prefix _PROB
PAYOFFS=	ACTION=	as OUTCOME=	Action names of final decision	Variables with prefix _ACT
	STATE=	as OUTCOME=	Outcome names	Variables with prefix _STAT
	VALUE=	N	Values of the scenario	Variables with prefix _VALU

[4] 'C' denotes character, 'N' denotes numeric, 'C/N' denotes character or numeric, and 'as X' denotes the same as variable X.

Variables in STAGEIN= Data Set

The following options specify the variables or variable lists in the STAGEIN= input data set that identify the stage name, its type, its outcomes, and the reward; and the immediate successor of each outcome for each stage in the decision model:

OUTCOME=(*variables*)

identifies all variables in the STAGEIN= data set that contain the outcome names of the stage specified by the STAGE= variable. If the OUTCOME= option is not specified, PROC DTREE looks for the default variable names that have the prefix _OUT in the data set. It is necessary to have at least one OUTCOME= variable in the STAGEIN= data set. The OUTCOME= variables can be either all character or all numeric. You cannot mix character and numeric variables as outcomes.

REWARD=(*variables*)

COST=(*variables*)

identifies all variables in the STAGEIN= data set that contain the reward for each outcome specified by the OUTCOME= variables. If the REWARD= option is not specified, PROC DTREE looks for the default variable names that have the prefix _REW in the data set. The number of REWARD= variables must be equal to the number of OUTCOME= variables in the data set. The REWARD= variables must have numeric values.

STAGE=*variable*

specifies the variable in the STAGEIN= data set that names the stages in the decision model. If the STAGE= option is omitted, PROC DTREE looks for the default variable named _STNAME_ in the data set. The STAGE= variable must be specified if the data set does not contain a variable named _STNAME_. The STAGE= variable can be either character or numeric.

SUCCESSOR=(*variables*)

SUCC=(*variables*)

identifies all variables in the STAGEIN= data set that contain the names of immediate successors (another stage) of each outcome specified by the OUTCOME= variables. These variables must be of the same type and length as those defined in the STAGE= option. If the SUCCESSOR= option is not specified, PROC DTREE looks for the default variable names that have the prefix _SUCC in the data set. The number of SUCCESSOR= variables must be equal to the number of OUTCOME= variables. The values of SUCCESSOR= variables must be stage names (values of STAGE= variables in the same data set).

TYPE=*variable*

identifies the variable in the STAGEIN= data set that contains the type identifier of the stage specified by the STAGE= variable. If the TYPE= option is omitted, PROC DTREE looks for the default variable named _STTYPE_ in the data set. The TYPE= variable must be specified if the data set does not contain a variable named _STTYPE_. The STAGE= variable can be either character or numeric.

The following are valid values for the TYPE= variable.

Value					Description
DECISION	or	D	or	1	Identifies the stage as a decision stage
CHANCE	or	C	or	2	Identifies the stage as an uncertain stage
END	or	E	or	3	Identifies the stage as an end stage

It is not necessary to specify an end stage in the STAGEIN= data set.

WEB=*variable*

HTML=*variable*

specifies the character variable in the STAGEIN= data set that identifies an HTML page for each stage. The procedure generates an HTML image map using this information for all the decision tree nodes corresponding to a stage.

Variables in PROBIN= Data Set

The following options specify the variables or variable lists in the PROBIN= input data set that identify the given outcome names, the event (outcome) name, and the conditional probability for each outcome of a chance stage.

EVENT=(*variables*)

identifies all variables in the PROBIN= data set that contain the names of events (outcomes) that probabilities depend on the outcomes specified by the GIVEN= variables. If the EVENT= option is not specified, PROC DTREE looks for the default variable names that have the prefix _EVEN in the data set. You must have at least one EVENT= variable in the PROB= data set. The values of EVENT= variables must be outcome names that are specified in the STAGEIN= data set.

GIVEN=(*variables*)

identifies all variables in the PROBIN= data set that contain the given condition (a list of outcome names) of a chance stage on which the probabilities of the outcome depend. If the GIVEN= option is not specified, PROC DTREE looks for the default variable names that have the prefix _GIVE in the data set. It is not necessary to have GIVEN= variables in the data set but if there are any, their values must be outcome names that are specified in the STAGEIN= data set.

PROB=(*variables*)
identifies all variables in the PROBIN= data set that contain the values of the conditional probability of each event specified by the EVENT= variables, given that the outcomes specified by the GIVEN= variables have occurred. If the PROB= option is not specified, PROC DTREE looks for the default variable names that have the prefix _PROB in the data set. The number of PROB= variables in the data set must be equal to the number of EVENT= variables. The PROB= variables must have numeric values between 0 and 1 inclusive.

Variables in PAYOFFS= Data Set

The following options specify the variables or variable lists in the PAYOFFS= input data set that identify the possible scenarios (a sequence of outcomes), the final outcome names, and the evaluating values (payoff) of combinations of scenarios and final outcomes.

ACTION=(*variables*)
identifies all variables in the PAYOFFS= data set that contain the name of the final outcome for each possible scenario. If the ACTION= option is not specified, PROC DTREE looks for the default variable names that have the prefix _ACT in the data set. It is not necessary to have any ACTION= variables in the PAYOFFS= data set, but if there are any, their values must be outcome names specified in the STAGEIN= data set.

STATE=(*variables*)
identifies all variables in the PAYOFFS= data set that contain the names of outcomes that identify a possible scenario (a sequence of outcomes or a path in the decision tree), or the names of outcomes which combine with every outcome specified by the ACTION= variables to identify a possible scenario. If the STATE= option is not specified, PROC DTREE looks for the default variable names that have the prefix _STAT in the data set. It is not necessary to have any STATE= variables in the PAYOFFS= data set, but if there are any, their values must be outcome names specified in the STAGEIN= data set.

VALUE=(*variables*)
PAYOFFS=(*variables*)
UTILITY=(*variables*)
LOSS=(*variables*)
identifies all variables in the PAYOFFS= data set that contain the evaluating values or payoffs for all possible scenarios identified by the outcomes specified by the STATE= variables and the outcomes specified by the associated ACTION= variables. If the VALUE= option is not specified, PROC DTREE looks for the default variable names that have the prefix _VALU in the data set. The number of VALUE= variables must be equal to the number of ACTION= variables if there are any ACTION= variables. If there are no ACTION= variables in the data set, at least one STATE= variable must be in the data set, and the number of VALUE= variables must be exactly 1. The VALUE= variables must have numeric values.

VPC Statement

VPC *chance-stage-name* **;**

The VPC statement causes PROC TREE to compute the value of perfect control (the value of controlling an uncertainty). The effect of perfect control is that you can pick the outcome of an uncertain stage. This value gives an upper limit on the amount you should be willing to spend on any control procedure. Only the name of a chance stage can be used to calculate the value of perfect control. The procedure evaluates the decision tree, if it has not already done so, before computing this value.

VPI Statement

VPI *chance-stage-name* **;**

The VPI statement causes PROC DTREE to compute the value of perfect information. The value of perfect information is the benefit of resolving an uncertain stage before making a decision. This value is the upper limit on the improvement that can be expected for any information gathering effort. Only the name of a chance stage can be used to calculate the value of perfect information. The procedure evaluates the decision tree, if it has not already done so, before computing this value.

Details: DTREE Procedure

Input Data Sets

A decision problem is normally constructed in three steps:

1. A structuring of the problem in terms of decisions, uncertainties, and consequences.
2. Assessment of probabilities for the events.
3. Assessment of values (payoffs, losses, or preferences) for each consequence or scenario.

PROC DTREE represents these three steps in three SAS data sets. The STAGEIN= data set describes the structure of the problem. In this data set, you define all decisions and define all key uncertainties. This data set also contains the relative order of when decisions are made and uncertainties are resolved (planning horizon). The PROBIN= data set assigns probabilities for the uncertain events, and the PAYOFFS= data set contains the values (or utility measure) for each consequence or scenario. See the section "Overview: DTREE Procedure" on page 384 and the section "Getting Started: DTREE Procedure" on page 385 for a description of these three data sets.

PROC DTREE is designed to minimize the rules for describing a problem. For example, the PROBIN= data set is required only when the evaluation and analysis of a decision problem is necessary. Similarly, if the PAYOFFS= data set is not specified, the DTREE procedure assumes all payoff values are 0. The order of the observations is not important in any of the input data sets. Since a decision problem can be structured in many different ways and the data format is so flexible, all possible ways of describing a given decision problem cannot be shown here. However, some alternate ways of supplying the same problem are demonstrated. For example, the following statements show another way to input the oil wildcatter's problem described in the section "Introductory Example" on page 385.

```
data Dtoils3;
   format _STNAME_ $12. _STTYPE_ $2. _OUTCOM_ $10.
         _REWARD_ dollar12.0  _SUCCES_ $12.;
   input _STNAME_ $12. _STTYPE_  $4. _OUTCOM_ $12.
         _REWARD_ dollar12.0  _SUCCES_ $12.;
   datalines;
Drill       D   Drill              .    Cost
.           .   Not_drill          .    .
Cost        C   Low          -$150,000 Oil_deposit
.           .   Fair         -$300,000 Oil_deposit
.           .   High         -$500,000 Oil_deposit
Oil_deposit C   Dry                .    .
.           .   Wet            $700,000 .
.           .   Soaking      $1,200,000 .
;

/* -- create PAYOFFS= data set                    -- */
data Dtoilp3;
   input _EVENT1 $ _PROB1 _EVENT2 $ _PROB2;
   datalines;
Low    0.2   Dry       0.5
Fair   0.6   Wet       0.3
High   0.2   Soaking   0.2
;

/* -- PROC DTREE statements                       -- */
title "Oil Wildcatter's Problem";
proc dtree stagein=Dtoils3
           probin=Dtoilp3
           nowarning;
   evaluate / summary;
```

Note that the STAGEIN= data set describes the problem structure and the payoffs (using the REWARD= variable). Thus, the PAYOFFS= data set is no longer needed. Note also the changes made to the PROBIN= data set. The results, shown in Figure 7.8, are the same as those shown in Figure 7.2. However, the rewards and the payoffs are entirely different entities in decision tree models. Recall that the reward of an outcome means the *instant returns* when the *outcome* is realized. On the other hand, the payoffs are the *return* from each *scenario*. In the other words, the decision tree model described in the previous code and the model described in the section “Introductory Example” on page 385 are not equivalent, even though they have the same optimal decision.

Figure 7.8 Optimal Decision Summary of the Oil Wildcatter's Problem

Oil Wildcatter's Problem

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Drill	Decision
Cost	Chance
Oil_deposit	Chance
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Expected Value (MAXEV)
Optimal Decision Yields:	140000

Optimal Decision Policy
Up to Stage Drill

Alternatives or Outcomes	Cumulative Reward	Evaluating Value
Drill	$0	140000*
Not_drill	$0	0

You can try many alternative ways to specify your decision problem. Then you can choose the model that is most convenient and closest to your real problem. If PROC DTREE cannot interpret the input data, it writes a message to that effect to the SAS log unless the NOWARNING option is specified. However, there are mistakes that PROC DTREE cannot detect. These often occur after the model has been modified with either the MOVE statement or the MODIFY statement. After a MOVE statement is specified, it is a good idea to display the decision tree (using the TREEPLOT statement) and check the probabilities and value assessments to make sure they are reasonable.

For example, using the REWARD= variable in the STAGEIN= data set to input the payoff information as shown in the previous code may cause problems if you change the order of the stages. Suppose you move the stage `'Cost'` to the beginning of the tree, as was done in the section "Sensitivity Analysis and Value of Perfect Information" on page 391:

```
move Cost before Drill;
evaluate / summary;
```

The optimal decision yields $140,000, as shown on the optimal decision summary in Figure 7.9.

Figure 7.9 Optimal Decision Summary of the Oil Wildcatter's Problem

Oil Wildcatter's Problem

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Cost	Chance
Drill	Decision
Oil_deposit	Chance
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Expected Value (MAXEV)
Optimal Decision Yields:	140000

Optimal Decision Policy Up to Stage Drill			
Alternatives or Outcomes		**Cumulative Reward**	**Evaluating Value**
Low	**Drill**	$-150,000	450000*
Low	**Not_drill**	$-150,000	0
Fair	**Drill**	$-300,000	450000*
Fair	**Not_drill**	$-300,000	0
High	**Drill**	$-500,000	450000*
High	**Not_drill**	$-500,000	0

Recall that when this was done in the section "Sensitivity Analysis and Value of Perfect Information" on page 391, the optimal decision yielded $150,000. The reason for this discrepancy is that the cost of drilling, implemented as (negative) instant rewards here, is imposed on all scenarios including those that contain the outcome `'Not_drill'`. This mistake can be observed easily from the `Cumulative Reward` column of the optimal decision summary shown Figure 7.9.

Changing a decision stage to a chance stage is another example where using the MODIFY statement without care may cause problems. PROC DTREE cannot determine the probabilities of outcomes for this new chance stage unless they are included in the PROBIN= data set. In contrast to changing a chance stage to a decision stage (which yields insight on the value of gaining control of an uncertainty), changing a decision stage to a chance stage is not likely to yield any valuable insight even if the needed probability data are included in the PROBIN= data set, and it should be avoided.

Missing Values

In the STAGEIN= data set, missing values are allowed only for the STAGE= and TYPE= variables when the information of a stage is specified in more than one observation. In this case, missing values for the STAGE= and TYPE= variables are not allowed for the first observation defining the stage. Missing values for the OUTCOME=, GIVEN=, EVENT=, STATE=, and ACTION= variables are ignored. Missing values for the REWARD=, PROB=, and VALUE= variables are treated as 0. Missing values for the SUCCESSOR= variables are ignored if the value for the corresponding OUTCOME= variable is also missing.

Interactivity

The DTREE procedure is interactive. You start the procedure with the PROC DTREE statement and terminate it with the QUIT statement. It is not necessary to have a VARIABLES statement, although if you do include one, it must appear immediately after the PROC DTREE statement. The other statements such as the EVALUATE, MODIFY, MOVE, RECALL, RESET, SAVE, SUMMARY, TREEPLOT, VPC, and VPI, as well as the FOOTNOTE, GOPTIONS, NOTE, SYMBOL, and TITLE statements of SAS/GRAPH Software can be used in any order and as often as needed. One exception is that the RECALL statement has to be preceded by at least one SAVE statement.

When an error is detected during processing a statement other than the PROC DTREE statement and the QUIT statement, the procedure terminates if the option ERRHANDLE=QUIT is specified; otherwise it stops processing the current statement and waits for the next statement. In either case, an error message is written to the SAS log. If an error is detected in the PROC DTREE statement or the QUIT statement, the procedure terminates immediately with an error message.

Options in Multiple Statements

Many options that can be specified in the PROC DTREE statement can also appear in other statements. The options specified in the PROC DTREE statement remain in effect for all statements until the end of processing or until they are changed by a RESET statement. In this sense, those options are *global* options. The options specified in other statements are in effect only for the statement in which they are specified; hence, they are *local* options. If an option is specified both in the PROC DTREE statement and in another statement, the local specification overrides the global specification.

For example, the following statements

```
reset criterion=maxev;
evaluate / criterion=maxce rt=700000;
summary;
```

imply that the decision problem is evaluated and the optimal decision is determined based on the criterion MAXCE with RT=700000. However, the optimal decision summary produced by the SUMMARY statement is based on the option CRITERION=MAXEV and not the MAXCE criterion. If you want an option to be set permanently, use the RESET statement.

The Order of Stages

The order of stages is an important issue in structuring the decision problem. This sets the sequence of events or a time horizon and determines when a decision has to be made and when a chance stage has its uncertainty resolved. If a decision stage precedes another decision stage in the stages order, the decision to the right is made after the decision to the left. Moreover, the choice made in the first decision is remembered by the decision maker when he or she makes the second decision. Any chance stages that occur to the left of a decision stage have their uncertainty resolved before the decision is made. In other words, the decision maker knows what actually happened when he or she makes the decision. However, the order of two chance stages is fairly arbitrary if there are no other decision stages between them. For example, you can change the order of stages **`'Cost'`** and **`'Oil_Deposit'`** in the oil wildcatter's problem without affecting the results.

PROC DTREE determines the order (from left to right) of all stages specified in the STAGEIN= data set. The ordering is based on the rule that if stage **`A`** is the successor of an outcome of stage **`B`**, then stage **`A`** should occur to the right of (or after) stage **`B`**. With the MOVE statement, you can change this order. The MOVE statement is very useful in determining the value (benefit or penalty) of postponing or hurrying a decision. In particular, the *value of perfect information* about an uncertainty can be determined by moving the corresponding chance stage to the beginning. However, as mentioned in early sections, the results may be misleading if you use the MOVE statement without care. See the section "Input Data Sets" on page 420 for an example.

Suggestions for preventing misleading results are as follows:

- Using the SAVE statement, always save the original structure before making any changes.
- Use the TREEPLOT statement to display the complete decision tree and check all details after you change the order.

Evaluation

The EVALUATE statement causes PROC DTREE to calculate the optimal decision. The evaluate process is done by successive use of two devices:

- Find a certain equivalent for the uncertain evaluating values at each chance node.
- Choose the best alternative at each decision node.

The *certain equivalent* of an uncertainty is the certain amount you would accept in exchange for the uncertain venture. In other words, it is a single number that characterizes an uncertainty described by a probability distribution. This value is subjective and can vary widely from person to person. There are two quantities, closely related to the certain equivalent, that are commonly used by decision-makers: the most likely value and the expected value. The *most likely value* of an uncertainty is the value with the largest probability. The *expected value* is the sum of all outcomes multiplied by their probabilities.

Perhaps the most popular way to find the certain equivalent for an uncertainty is the use of *utility function* or *utility curve*. *Utility* is a measurement of relative *preference* to the decision maker for particular outcomes. The utility function assigns a utility to payoff when it is in terms of continuous values such as money. The certain equivalent of an uncertainty (a random variable) is calculated by the following steps:

1. Use the utility function or the utility curve to find the utility values of the outcomes.
2. Calculate the expected utility of the uncertainty.

3. Determine the certain equivalent of the uncertainty as the value that corresponding utility value is the expected utility.

Refer to Raiffa (1970) for a complete discussion of the utility function.

A simple case that is commonly used is the straight line utility curve or the linear utility function. The linear utility function has the form

$$u(x) = a + bx$$

where x is the evaluating value, and a and b are parameters set by the choice of two points in the utility curve. For example, if the utility curve passes two points $u(0) = 0$ and $u(1000) = 1$, then parameters a and b are set by $a = 0$ and $b = 1 / 1000$. The certain equivalent of an uncertainty based on this function is the expected value.

Another special case that is commonly used is the exponential utility function, as

$$u(x) = a - b \times \exp(-x/r)$$

where, again, a and b can be set by the choice of two arbitrary points in the utility curve. For example, if your utility curve goes through points (0,0) and (1000,1), then a and b are given by

$$a = b = 1/[1 - \exp(-1000/r)]$$

If an uncertain venture A has n events, event i having probability p_i and payoff x_i, and if the utility function is an exponential function as in the preceding example, then the certain equivalent of A is

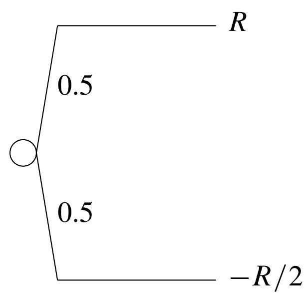
$$\mathrm{CE}(A) = -r \ln \left[\sum_{i=1}^{n} p_i \exp(-x_i/r) \right]$$

and is independent of the choice of values for a and b (provided that $b > 0$) (Raiffa 1970).

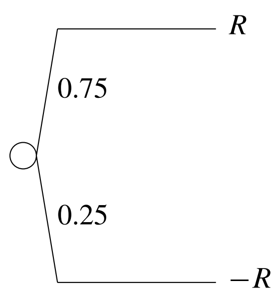
The parameter r, called the *risk tolerance*, describes the curvature of the utility function. Moreover, the quantity $1/r$, called *risk aversion coefficient* (Howard 1968) is a measure of risk aversion.

Experimental results show that within a reasonable range of values, many utility curves can be fit quite well by an exponential function.

If your utility function is an exponential function as in the preceding example, the risk tolerance can be estimated by the largest number R for which the following venture is still acceptable to you.



A similar way to approximate the risk tolerance is to find the largest value R for which the venture is acceptable (Howard 1988).



For corporate decision making, there are some rules of thumb for estimating the risk tolerance. Examples are to set risk tolerance about equal to one of the following:

- net income of the company
- one sixth of equity
- six percent of net sales

To reveal how well these rules perform in assessing corporate risk tolerance, Howard (1988) provided the following two tables: Table 7.12 shows the relationship between the risk tolerance and financial measures of four large oil and chemicals companies. There, the risk tolerances are obtained from the top executives of the companies. The net sales, net income, and equity are obtained from the annual reports of the four companies.

Table 7.12 Relating Corporate Risk Tolerance to Financial Measures

Measure ($ millions)	Company A	B	C	D
Net Sales	2,300	3,307	16,000	31,000
Net Income	120	152	700	1,900
Equity	1,000	1,153	6,500	12,000
Risk Tolerance	150	200	1,000	2,000

Table 7.13 shows the ratio of risk tolerance to each of the other quantities.

Table 7.13 Ratios of Corporate Risk Tolerance to Financial Measures

Measure	Company A	B	C	D	Average
RT/Sales	0.0652	0.0605	0.0625	0.0645	0.0632
RT/Income	1.25	1.32	1.43	1.05	1.26
RT/Equity	0.150	0.174	0.154	0.167	0.161

Once the certain equivalents for all chance nodes are assessed, the choice process at each decision node is fairly simple; select the alternative yielding either the maximum or the minimum (depending on the problem) future certain equivalent value.* You can use the CRITERION= option to control the way the certain equivalent is calculated for each chance node and the optimal alternative is chosen at each decision node. Possible values for the CRITERION= option are listed in Table 7.5. If you use an exponential utility function, the RT= option can be used to specify your risk tolerance. You also have control over how to present the solution. By default, PROC DTREE writes the value of the optimal decisions to the SAS log. In addition, with the SUMMARY option, you can ask PROC DTREE to display the optimal decision summary to the output.

Displayed Output

The SUMMARY statement and the SUMMARY option in an EVALUATE statement cause PROC DTREE to display a optimal decision summary for the decision model. This output is organized into various tables, and they are discussed in order of appearance.

[1]The future certain equivalent value is often referred to as the evaluating value in this documentation.

Order of stages

The Order of stages table lists all stages, and their types, in order of appearance in the decision model. See the section "The Order of Stages" on page 424 for details.

For ODS purposes, the label of the Order of stages table is "Stages."

Decision Parameters

The Decision Parameters table describes the criterion used for determining the optimal decision and the certain equivalent for replacing uncertainties. If you specify the option CRITERION=MAXCE or CRITERION=MINCE in the PROC DTREE statement or in the EVALUATE statement, an additional row is added to the table listing the value of the risk tolerance. It also contains a row showing the value of the optimal decision yields. For additional information, see the section "Evaluation" on page 425.

For ODS purposes, the label of the Decision Parameters table is "Parameters."

Optimal Decision Policy

By default, PROC DTREE produces an Optimal Decision Policy table for each decision stage. You can use the TARGET= option to force PROC DTREE to produce only one table for a particular stage. The Alternatives or Outcomes columns list the events in the scenario that leads to the current stage. The Cumulative Reward column lists the rewards accumulated along the scenario to the events of the current target stage. The Evaluating Value column lists the values that can be expected form the events of the target stage. An asterisk (*) is placed beside an evaluating value indicates the current event is the best alternative of the given scenario.

For ODS purposes, the label of the Optimal Decision Policy table is "Policy."

Displaying the Decision Tree

PROC DTREE draws the decision tree either in line-printer mode or in graphics mode. However, you need to have SAS/GRAPH software licensed at your site to use graphics mode. In many cases, the procedure draws the decision tree across page boundaries. If the decision tree diagram is drawn on multiple pages, the procedure numbers each page of the diagram on the upper right corner of the page (unless the NOPAGENUM option is specified). The pages are numbered starting with the upper left corner of the entire diagram. Thus, if the decision tree diagram is broken into three horizontal and four vertical levels and you want to paste all the pieces together to form one picture, they should be arranged as shown in Figure 7.10.

Figure 7.10 Page Layout of the Decision Tree Diagram

1	5	9
2	6	10
3	7	11
4	8	12

The number of pages that are produced depends on the size of the tree and on the number of print positions that are available in the horizontal and vertical directions. Table 7.14 lists all options you can use to control the number of pages.

Table 7.14 Options That Control the Number of Pages

Option	Effect
DISPLAY=	Amounts of information displayed on the diagram
MAXPREC=	Maximum decimal width allowed (the precision) to format numerical values into $w.d$ format
MAXWIDTH=	Maximum field width allowed to format numerical values
NOLABEL	No labels are displayed on the diagram
NWIDTH=	Maximum field width allowed to format outcome names
YBETWEEN=	Vertical spaces between two successive end nodes

If the GRAPHICS option is used, the following options can be used to control the number of pages:

- The COMPRESS option draws the entire decision tree on one page.
- The HSYMBOL= option controls the height of all symbols.
- The HTEXT= option controls the height of text in the tree.
- The HEIGHT= option in a SYMBOL definition specifies the height of a symbol.
- The HTEXT= option in a GOPTIONS statement specifies the height of all text.
- The HTITLE= option in a GOPTIONS statement specifies the height of the first title line.
- The HPOS= and VPOS= options in a GOPTIONS statement change the number of rows and columns.

The font used for all text can also affect the number of pages needed. Some fonts take more space than others.

If the decision tree diagram is produced on a line printer, you can use the FORMCHAR= option to control the appearance of the links and the junctions of the diagram. When you specify the GRAPHICS option, several options are available to enhance the appearance of the decision tree diagram. These are described in the section "Graphics Options" on page 405. In addition, many other options are available in the GOPTIONS and SYMBOL statements for controlling the details of graphics output. See the relevant chapters in *SAS/GRAPH Software: Reference* for a detailed discussion of the GOPTIONS and SYMBOL statements. ODS graphical styles can be applied to the decision tree diagram; see the section "ODS Style Templates" on page 434 for more information.

Table 7.15, Table 7.16, and Table 7.17 show the relationship among the options for controlling the appearance of texts, nodes, and links, respectively. The order that PROC DTREE uses in determining which option is in effect is also provided. The order assumes that the GSTYLE system option is in effect; if that is not the case, then the steps that refer to ODS style templates are ignored. Names with arguments indicate style elements and attributes of the current ODS style template. For example, "GraphDataText('Font')" refers to the Font attribute of the GraphDataText element.

For ODS purposes, the label of the decision tree diagram drawn in line-printer quality is "Treeplot."

Table 7.15 Options That Control Text Appearance

Object	Specification		Search Order
Text	Font	1.	The FTEXT= option
		2.	The FTEXT= option in a GOPTIONS statement
		3.	GraphDataText('Font')
		4.	Hardware font
	Color	1.	The CTEXT= option
		2.	The CTEXT= option in a GOPTIONS statement
		3.	GraphDataText('ContrastColor')
		4.	The first color in the colors list
	Height	1.	The value of the HTEXT= option[1] times the value of the HTEXT= option[2] in a GOPTIONS statement

[1] If this option is not specified, the default value 1 is used.

[2] The default value of this option is 1 unit.

Table 7.16 Options That Control Node Appearance

Object	Specification		Search Order
Chance Nodes	Symbol	1.	The VSYMBOLC= option
		2.	The VALUE= and FONT= options in the *m*th generated SYMBOL definition, if SYMBOLC=*m* is specified
		3.	The default symbol, CIRCLE
	Color	1.	The CSYMBOLC= option
		2.	The CV= option in the *m*th generated SYMBOL definition, if SYMBOLC=*m* is specified
		3.	The CSYMBOL= option in a GOPTIONS statement
		4.	GraphData1('ContrastColor')
		5.	The fifth color in the colors list

Table 7.16 (continued)

Object	Specification		Search Order
	Height	1.	*h* times the value of the HEIGHT= option in the *m*th generated SYMBOL definition, if both HSYMBOL=*h* and SYMBOLC=*m* are specified
		2.	The HSYMBOL= option, if it is specified
		3.	The HEIGHT= option in the *m*th generated symbol definition, if SYMBOLC=*m* is specified
		4.	The default value, 1 cell
Decision Nodes	Symbol	1.	The VSYMBOLD= option
		2.	The VALUE= and FONT= options in the *d*th generated SYMBOL definition, if SYMBOLD=*d* is specified
		3.	The default value, SQUARE
	Color	1.	The CSYMBOLD= option
		2.	The CV= option in the *d*th generated SYMBOL definition, if SYMBOLD=*d* is specified
		3.	The CSYMBOL= option in a GOPTIONS statement
		4.	GraphData5('ContrastColor')
		5.	The fourth color in the colors list
	Height	1.	*h* times the value of the HEIGHT= option in the *d*th generated SYMBOL definition, if both HSYMBOL=*h* and SYMBOLD=*d* are specified
		2.	The HSYMBOL= option, if it is specified
		3.	The HEIGHT= option in the *d*th generated symbol definition, if SYMBOLD=*d* is specified
		4.	The default value, 1 cell
End Nodes	Symbol	1.	The VSYMBOLE= option
		2.	The VALUE= and FONT= options in the *n*th generated SYMBOL definition, if SYMBOLE=*n* is specified
		3.	The default value, DOT
	Color	1.	The CSYMBOLE= option
		2.	The CV= option in the *n*th generated SYMBOL definition if the option SYMBOLE=*n* is specified
		3.	The CSYMBOL= option in a GOPTIONS statement
		4.	GraphData8('ContrastColor')
		5.	The sixth color in the colors list
	Height	1.	*h* times the value of the HEIGHT= option in the *n*th generated SYMBOL definition, if both HSYMBOL=*h* and SYMBOLE=*n* are specified
		2.	The HSYMBOL= option, if it is specified
		3.	The HEIGHT= option in the *n*th generated symbol definition, if SYMBOLE=*n* is specified
		4.	The default value, 1 cell

Table 7.17 Options That Control Link Appearance

Object	Specification		Search Order
Links for Regular Outcomes	Type	1.	The LSTYLE= option
		2.	The LINE= option in the *i*th generated SYMBOL definition, if LINKA=*i* is specified
		3.	The default value, 1 (solid line)
	Color	1.	The CLINK= option

Table 7.17 (continued)

Object	Specification		Search Order
		2.	The CI= option in the *i*th generated SYMBOL definition, if LINKA=*i* is specified
		3.	GraphData3('ContrastColor')
		4.	The third color in the colors list
	Thickness	1.	The LWIDTH= option
		2.	The WIDTH= option in the *i*th generated SYMBOL definition, if LINKA=*i* is specified
		3.	The default value, 1
Links for Optimal Decision	Type	1.	The LSTYLEB= option
		2.	The LINE= option in the *j*th generated SYMBOL definition, if LINKB=*j* is specified
		3.	The default value, 1 (solid line)
	Color	1.	The CBEST= option
		2.	The CI= option in the *j*th generated SYMBOL definition, if LINKB=*j* is specified
		3.	GraphData2('ContrastColor')
		4.	The second color in the colors list
	Thickness	1.	The LWIDTHB= option
		2.	The WIDTH= option in the *j*th generated SYMBOL definition, if LINKB=*j* is specified
		3.	2 times the thickness of links that represent regular outcomes
Links That Fall across Pages	Type	1.	The LSTYLEC= option
		2.	The LINE= option in the *k*th generated SYMBOL definition, if LINKC=*k* is specified
		3.	The default value, 2 (dot line)
	Color	1.	Depends on whether or not it represents an optimal decision
	Thickness	1.	Depends on whether or not it represents an optimal decision

Web-Enabled Decision Tree

The WEB= variable in the STAGEIN= data set enables you to define an HTML reference for each stage. This HTML reference is currently associated with all the decision tree nodes that correspond to the stage. The WEB= variable is a character variable, and the values need to be of the form HREF=*htmlpage*.

In addition, you can also store the coordinate and link information defined by the WEB= option in a SAS data set by specifying the IMAGEMAP= option in the PROC DTREE statement or in the TREEPLOT statement. If you process this SAS data set by using a DATA step, you can generate customized HTML pages for your decision tree diagram.

ODS Table Names

PROC DTREE assigns a name to each table that it creates. You can use these names to reference the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 7.18. For more information about ODS, see the chapter on ODS in the *SAS/STAT User's Guide*.

Table 7.18 ODS Tables Produced in PROC DTREE

ODS Table Name	Description	Statement / Option
Parameters	Decision parameters	SUMMARY or EVALUATE / SUMMARY
Policy	Optimal decision policy	SUMMARY or EVALUATE / SUMMARY
Stages	List of stages in order	SUMMARY or EVALUATE / SUMMARY
Treeplot	Line-printer plot of decision tree	TREEPLOT / LINEPRINTER

ODS Style Templates

ODS style templates, or *styles*, control the overall look of your output. An ODS style template consists of a set of *style elements*. A style element is a collection of *style attributes* that apply to a particular feature or aspect of the output. You can specify a value for each attribute in a style. See Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*), for a thorough discussion of ODS Graphics.

To create your own style or to modify a style for use with ODS Graphics, you need to understand the relationships between style elements and graph features. This information is provided in the ODS Graphics documentation at **http://support.sas.com/documentation/onlinedoc/base/**. You create and modify style templates with the TEMPLATE procedure. For more information, see the section "TEMPLATE Procedure: Creating a Style Template" in the *SAS Output Delivery System: User's Guide*. Kuhfeld (2010) also offers detailed information and examples.

PROC DTREE Style Template

A predefined ODS style template named DTREE is available for the DTREE procedure. You can use the template to maintain a consistent appearance in all graphical output produced by the procedure.

To change the current style, specify the STYLE= option in an ODS destination statement. The specified style is applied to all output for that destination until you change or close the destination or start a new SAS session. For example, the following statement specifies that ODS should apply the DTREE style template to all HTML output:

```
ods html style=dtree;
```

To disable the use of graphical styles, specify the SAS system option NOGSTYLE.

The parent style template for the DTREE style is the DEFAULT style. Table 7.19 lists the style elements (in bold) and corresponding attributes specified in the DTREE style. The table also indicates which, if any, PROC DTREE options or graphics options (in a GOPTIONS statement) can be used to override the value of a style attribute.

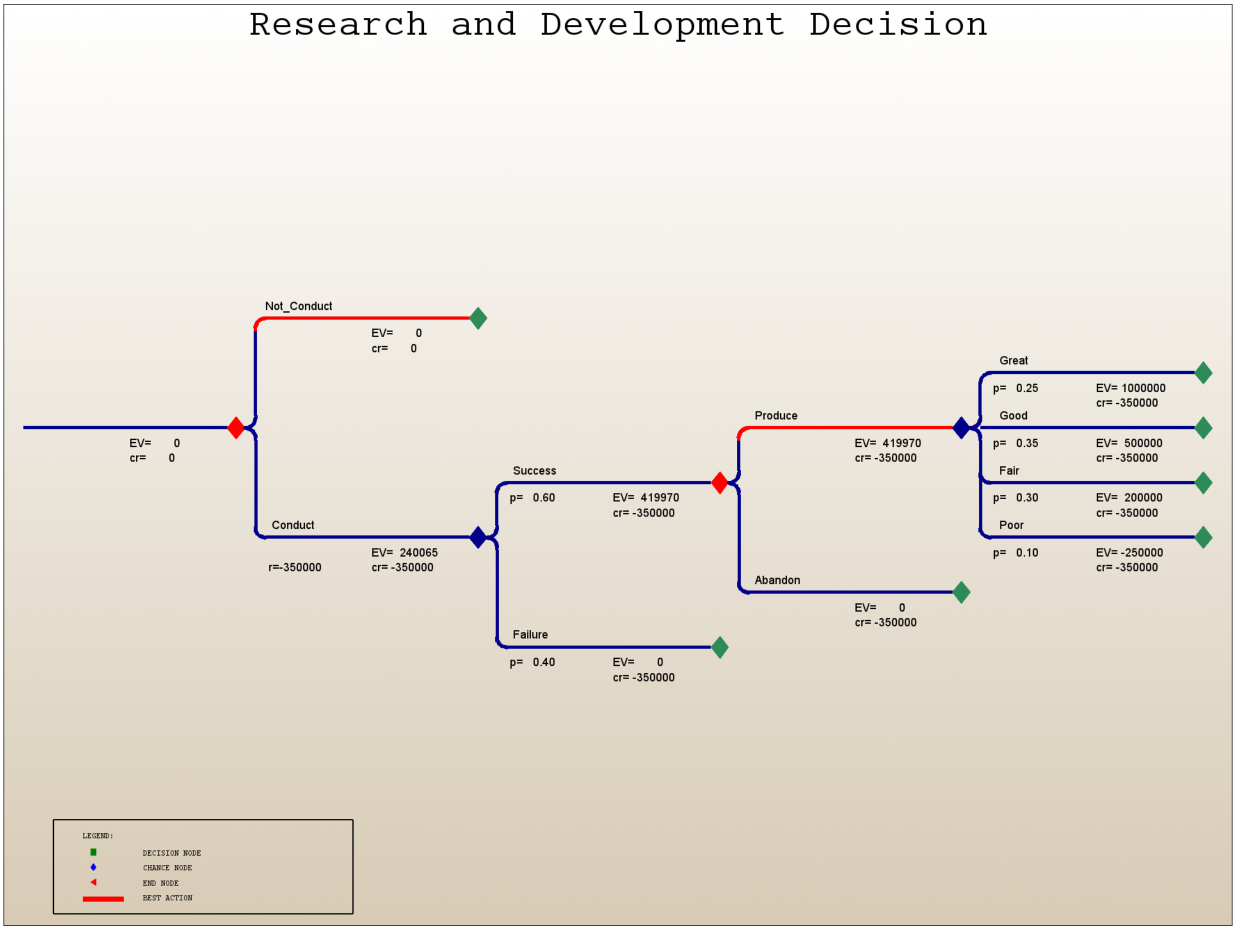
Table 7.19 Style Elements and Attributes in the DTREE Style

Element/Attributes	Description	DTREE Option	GOPTION
GraphColors	Colors of various graph features		
gcdata5	Decision nodes	CSYMBOLD=	COLORS=
gcdata1	Chance nodes	CSYMBOLC=	COLORS=
gcdata8	End nodes	CSYMBOLE=	COLORS=
gcdata3	Regular links	CLINK=	COLORS=
gcdata2	Optimal links	CBEST=	COLORS=
gtextt	Title text		CTITLE=
gtext	Text		CTEXT=
GraphFonts	Fonts for various graph features		
GraphDataFont	Default		FTEXT=
GraphLabelFont	Labels		FTEXT=
GraphTitleFont	Title text		FTITLE=
GraphDataText	Attributes related to general text		
ContrastColor	GraphColors('gtext')	CTEXT=	CTEXT=
Font	GraphFonts('GraphDataFont')	FTEXT=	FTEXT=
GraphTitleText	Attributes related to title text		
Color	GraphColors('gtext')		CTITLE=
Font	GraphFonts('gtext')		FTITLE=
GraphTitle1Text	Attributes related to the first title text		
Color	GraphColors('gtext')		CTITLE=
Font	GraphFonts('gtext')		FTITLE=
GraphLabelText	Attributes related to label text		
Color	GraphColors('glabel')		CTEXT=
Font	GraphFonts('GraphTitleFont')		FTEXT=
GraphDataDefault	Default values		
Color	GraphColors('gdata')		COLORS=
GraphBackground	Attributes related to graph background		
Image	Dtree.jpg		CBACK=

Attributes that you do not override retain the values specified in the style template.

Figure 7.11 demonstrates features of the DTREE graphical style. The decision tree in the figure is the first output from Example 7.4.

Figure 7.11 DTREE Style Template: Example



Default Values

If the SAS system option GSTYLE is in effect (this is the default), then the default values of certain PROC DTREE options can depend on the current ODS style template. See the section "Displaying the Decision Tree" on page 429 for more information.

Precision Errors

When PROC DTREE detects an error, it displays a message on the SAS log to call it to your attention. If the error is in a statement other than the PROC DTREE statement and the QUIT statement, and if the ERRHANDLE=QUIT option is not specified, the procedure ignores the erroneous statement and waits for you to enter another statement. This gives you a chance to correct the mistake you made and keep running. You can exit the procedure at any time by specifying the QUIT statement.

If the error is in an input data set, typically, you will have to edit the data set and then reinvoke PROC DTREE. In one case, however, you can use an option to correct the problem. You may receive an error message indicating that the sum of probabilities for a particular chance stage does not equal 1.0. If it is caused by roundoff errors in the summation, then you can reset the TOLERANCE= option to correct this error. For example, suppose that your problem contains a chance stage that has three outcomes, `'Out1'`, `'Out2'` and `'Out3'`, and each has probability 1/3. Suppose also that you input their probabilities in the PROBIN= data set as follows:

```
Out1    Out2    Out3    0.3333  0.3333  0.3333
```

Then, PROC DTREE detects the total probabilities for that stage as 0.9999, not equal to 1, and hence displays an error message. The following RESET statement fixes the error:

```
reset tolerance=0.00015;
```

Alternatively, you can specify the AUTOSCALE option to ask the procedure to rescale the probabilities whenever this situation occurs.

Computer Resource Requirements

There is no inherent limit on the size of the decision tree model that can be evaluated and analyzed with the DTREE procedure. The numbers of stages and outcomes are constrained only by the amount of memory available. Naturally, there needs to be a sufficient amount of core memory available in order to invoke and initialize the SAS system. Furthermore, more memory is required to load the graphics sublibrary if the GRAPHICS option is specified. As far as possible, the procedure attempts to store all the data in core memory. However, if the problem is too large to fit in core memory, the procedure resorts to the use of utility data sets and swaps between core memory and utility data sets as necessary.

The storage requirement for the data area required by the procedure is proportional to the number of stages and outcomes as well as the number of nodes[2] in the decision tree model. The time required depends heavily on the number of nodes in the decision tree.

Examples: DTREE Procedure

This section contains six examples that illustrate several features and applications of the DTREE procedure. The aim of this section is to show you how to use PROC DTREE to solve your decision problem and gain valuable insight into its structure.

Example 7.1 and Example 7.2 show two methods frequently used to spread the risk of a venture: buy insurance and enter a partnership. Example 7.1 also illustrates the use of the VARIABLE statement to identify the variables in the input data sets. Example 7.3 illustrates the use of the graphics options to produce a graphics quality decision tree diagram. Example 7.4 illustrates the use of SYMBOL and GOPTIONS statements and the Annotate facility to control the appearance of the decision tree diagram. Example 7.5 demonstrates an application of PROC DTREE for financial decision problems. It also illustrates a situation where redundant data are necessary to determine the value of information. In addition, it shows a case where the results from the VPI and VPC statements are misleading if they are used without care. Example 7.6

[2]The number of nodes depends on the number of stages and the number of outcomes for each stage.

shows an application in litigation, a sophisticated use of sensitivity analysis. It also shows you how to deal with the value of future money.

Finally, Table 7.27 and Table 7.28 list all the examples in this chapter, and the options and statements in the DTREE procedure that are illustrated by each example.

Example 7.1: Oil Wildcatter's Problem with Insurance

Again consider the oil wildcatter's problem introduced in the section "Introductory Example" on page 385. Suppose that the wildcatter is concerned that the probability of a dry well may be as high as 0.5.

The wildcatter has learned that an insurance company is willing to offer him a policy that, with a premium of $130,000, will redeem $200,000 if the well is dry. He would like to include the alternative of buying insurance into his analysis. One way to do this is to include a stage for this decision in the model. The following DATA step reads this new decision problem into the STAGEIN= data set named Dtoils4. Notice the new stage named `'Insurance'`, which represents the decision of whether or not to buy the insurance. Also notice that the cost of the insurance is represented as a negative reward of $130,000.

```
/* -- create the STAGEIN= data set                  -- */
data Dtoils4;
format Stage $12. Stype $2. Outcome $14.
     Succ $12. Premium dollar12.0;
input Stage $12. Stype $4. Outcome $16. Succ $12.
    Premium dollar12.0;
datalines;
Drill        D   Drill           Insurance            .
.            .   Not_Drill       .                    .
Insurance    D   Buy_Insurance   Cost         -$130,000
.            .   Do_Not_Buy      Cost                 .
Cost         C   Low             Oil_Deposit          .
.            .   Fair            Oil_Deposit          .
.            .   High            Oil_Deposit          .
Oil_Deposit C    Dry             .                    .
.            .   Wet             .                    .
.            .   Soaking         .                    .
;
```

Probabilities associated with the uncertain events are given in the PROBIN= data set named Dtoilp4. Except for the order of the variables in this data set, it is the same as the Dtoilp1 data set given in the section "Introductory Example" on page 385.

```
/* -- create the PROBIN= data set                   -- */
data Dtoilp4;
input (V1-V3) ($) P1-P3 ;
datalines;
Low         Fair        High        0.2     0.6     0.2
Dry         Wet         Soaking     0.5     0.3     0.2
;
```

The payoffs for this problem are now calculated to include the cost and value of the insurance. The following DATA step does this.

```
/* -- create PAYOFFS= data set                         -- */
data Dtoilu4;
input (Cost Deposit Drill Insuran) ($16.) ;
format Drill $9. Insuran $14. Payoff dollar12.0;

/* determine the cost for this scenario */
if      Cost='Low'  then Rcost=150000;
else if Cost='Fair' then Rcost=300000;
else                     Rcost=500000;

/* determine the oil deposit and the corresponding  */
/* net payoff for this scenario                     */
if      Deposit='Dry' then Return=0;
else if Deposit='Wet' then Return=700000;
else                       Return=1200000;

 /* calculate the net return for this scenario */
if      Drill='Not_Drill' then Payoff=0;
else                           Payoff=Return-Rcost;

/* determine redeem received for this scenario */
if Insuran='Buy_Insurance' and Deposit='Dry' then
 Payoff=Payoff+200000;

/* drop unneeded variables */
drop Rcost Return;

datalines;
Low             Dry             Not_Drill       .
Low             Dry             Drill           Buy_Insurance
Low             Dry             Drill           Do_Not_Buy
Low             Wet             Not_Drill       .
Low             Wet             Drill           Buy_Insurance
Low             Wet             Drill           Do_Not_Buy
Low             Soaking         Not_Drill       .
Low             Soaking         Drill           Buy_Insurance
Low             Soaking         Drill           Do_Not_Buy
Fair            Dry             Not_Drill       .
Fair            Dry             Drill           Buy_Insurance
Fair            Dry             Drill           Do_Not_Buy
Fair            Wet             Not_Drill       .
Fair            Wet             Drill           Buy_Insurance
Fair            Wet             Drill           Do_Not_Buy
Fair            Soaking         Not_Drill       .
Fair            Soaking         Drill           Buy_Insurance
Fair            Soaking         Drill           Do_Not_Buy
High            Dry             Not_Drill       .
High            Dry             Drill           Buy_Insurance
High            Dry             Drill           Do_Not_Buy
High            Wet             Not_Drill       .
High            Wet             Drill           Buy_Insurance
High            Wet             Drill           Do_Not_Buy
High            Soaking         Not_Drill       .
High            Soaking         Drill           Buy_Insurance
High            Soaking         Drill           Do_Not_Buy
;
```

The payoff table can be displayed with the following PROC PRINT statement:

```
   /* -- print the payoff table          -- */
title "Oil Wildcatter's Problem";
title3 "The Payoffs";

proc print data=Dtoilu4;
run;
```

The table is shown in Output 7.1.1.

Output 7.1.1 Payoffs of the Oil Wildcatter's Problem with an Insurance Option

Oil Wildcatter's Problem

The Payoffs

Obs	Cost	Deposit	Drill	Insuran	Payoff
1	Low	Dry	Not_Drill		$0
2	Low	Dry	Drill	Buy_Insurance	$50,000
3	Low	Dry	Drill	Do_Not_Buy	$-150,000
4	Low	Wet	Not_Drill		$0
5	Low	Wet	Drill	Buy_Insurance	$550,000
6	Low	Wet	Drill	Do_Not_Buy	$550,000
7	Low	Soaking	Not_Drill		$0
8	Low	Soaking	Drill	Buy_Insurance	$1,050,000
9	Low	Soaking	Drill	Do_Not_Buy	$1,050,000
10	Fair	Dry	Not_Drill		$0
11	Fair	Dry	Drill	Buy_Insurance	$-100,000
12	Fair	Dry	Drill	Do_Not_Buy	$-300,000
13	Fair	Wet	Not_Drill		$0
14	Fair	Wet	Drill	Buy_Insurance	$400,000
15	Fair	Wet	Drill	Do_Not_Buy	$400,000
16	Fair	Soaking	Not_Drill		$0
17	Fair	Soaking	Drill	Buy_Insurance	$900,000
18	Fair	Soaking	Drill	Do_Not_Buy	$900,000
19	High	Dry	Not_Drill		$0
20	High	Dry	Drill	Buy_Insurance	$-300,000
21	High	Dry	Drill	Do_Not_Buy	$-500,000
22	High	Wet	Not_Drill		$0
23	High	Wet	Drill	Buy_Insurance	$200,000
24	High	Wet	Drill	Do_Not_Buy	$200,000
25	High	Soaking	Not_Drill		$0
26	High	Soaking	Drill	Buy_Insurance	$700,000
27	High	Soaking	Drill	Do_Not_Buy	$700,000

To find the optimal decision, call PROC DTREE with the following statements:

```
   /* -- PROC DTREE statements                       -- */
title "Oil Wildcatter's Problem";

proc dtree stagein=Dtoils4
```

```
            probin=Dtoilp4
            payoffs=Dtoilu4
            nowarning
            ;

   variables / stage=Stage type=Stype outcome=(Outcome)
               reward=(Premium) successor=(Succ)
               event=(V1 V2 V3) prob=(P1 P2 P3)
               state=(Cost Deposit Drill Insuran)
               payoff=(Payoff);

    evaluate;
    summary / target=Insurance;
```

The VARIABLES statement identifies the variables in the input data sets. The yield of the optimal decision is written to the SAS log as:

```
NOTE: Present order of stages:

      Drill(D), Insurance(D), Cost(C), Oil_Deposit(C),
      _ENDST_(E).

NOTE: The currently optimal decision yields 140000.
```

The optimal decision summary produced by the SUMMARY statements are shown in Output 7.1.2. The summary in Output 7.1.2 shows that the insurance policy is worth \$240,000 - \$140,000 = \$100,000, but since it costs \$130,000, the wildcatter should reject such an insurance policy.

Output 7.1.2 Summary of the Oil Wildcatter's Problem

Oil Wildcatter's Problem

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Drill	Decision
Insurance	Decision
Cost	Chance
Oil_Deposit	Chance
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Expected Value (MAXEV)
Optimal Decision Yields:	\$140,000

Output 7.1.2 *continued*

Optimal Decision Policy Up to Stage Insurance			
Alternatives or Outcomes		Cumulative Reward	Evaluating Value
Drill	Buy_Insurance	$-130,000	$240,000
Drill	Do_Not_Buy	$0	$140,000*

Now assume that the oil wildcatter is risk averse and has an exponential utility function with a risk tolerance of $1,200,000. In order to evaluate his problem based on this decision criterion, the wildcatter reevaluates the problem with the following statements:

```
reset criterion=maxce rt=1200000;
summary / target=Insurance;
```

The output from PROC DTREE given in Output 7.1.3 shows that the decision to purchase an insurance policy is favorable in the risk-averse environment. Note that an EVALUATE statement is not necessary before the SUMMARY statement. PROC DTREE evaluates the decision tree automatically when the decision criterion has been changed using the RESET statement.

Output 7.1.3 Summary of the Oil Wildcatter's Problem with RT = 1,200,000

Oil Wildcatter's Problem

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	Type
Drill	Decision
Insurance	Decision
Cost	Chance
Oil_Deposit	Chance
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Certain Equivalent Value (MAXCE)
Risk Tolerance:	$1,200,000
Optimal Decision Yields:	$45,728

Optimal Decision Policy Up to Stage Insurance			
Alternatives or Outcomes		Cumulative Reward	Evaluating Value
Drill	Buy_Insurance	$-130,000	$175,728*
Drill	Do_Not_Buy	$0	$44,499

Example 7.2: Oil Wildcatter's Problem in Risk-Averse Setting

Continuing with the oil wildcatter's problem, suppose that in addition to possibly buying insurance to spread the risk of the venture, the wildcatter is considering sharing the risk by selling a portion of this venture to other investors. Now, the decision he faces is whether to buy insurance or not and what percentage of the investment to divest. Again, assume that the wildcatter is risk averse with a risk tolerance of $1,200,000. Notice that in the program that follows the `'Divestment'` decision includes possibilities of no divestment to 100% divestment in 10% increments.

```
/* -- create the STAGEIN= data set                      -- */
data Dtoils4;
format _STNAME_ $12. _OUTCOM_ $15. _SUCCES_ $12.;
input _STNAME_ $ _STTYPE_ $ _OUTCOM_ $
    _SUCCES_ $ ;
datalines;
Divestment      Decision    No_Divestment      Insurance
.               .           10%_Divestment     Insurance
.               .           20%_Divestment     Insurance
.               .           30%_Divestment     Insurance
.               .           40%_Divestment     Insurance
.               .           50%_Divestment     Insurance
.               .           60%_Divestment     Insurance
.               .           70%_Divestment     Insurance
.               .           80%_Divestment     Insurance
.               .           90%_Divestment     Insurance
.               .           100%_Divestment    .
Insurance       Decision    Buy_Insurance      Cost
.               .           Do_Not_Buy         Cost
Cost            Chance      Low                Oil_Deposit
.               .           Fair               Oil_Deposit
.               .           High               Oil_Deposit
Oil_Deposit     Chance      Dry                .
.               .           Wet                .
.               .           Soaking            .
;
```

The probabilities associated with the uncertain events are given in the PROBIN= data set named Dtoilp4. Except for the order of the variables in this data set, it is the same as the Dtoilp1 data set used in the section "Introductory Example" on page 385.

```
/* -- create the PROBIN= data set                              -- */
data Dtoilp4;
input _EVENT1 $ _PROB1  _EVENT3 $ _PROB3 ;
datalines;
Low         0.2     Dry         0.5
Fair        0.6     Wet         0.3
High        0.2     Soaking     0.2
;

/* -- create the PAYOFFS= data set                             -- */
data Dtoilu4(drop=i j k l);
length _STATE1-_STATE4 $16. ;
```

```
format _VALUE_ dollar12.0;

/* define and initialize arrays */
array DIVEST{11}  $16. _TEMPORARY_ ('No_Divestment',
                                     '10%_Divestment',
                                     '20%_Divestment',
                                     '30%_Divestment',
                                     '40%_Divestment',
                                     '50%_Divestment',
                                     '60%_Divestment',
                                     '70%_Divestment',
                                     '80%_Divestment',
                                     '90%_Divestment',
                                     '100%_Divestment'  );
array INSUR{3} $16.    _TEMPORARY_ ('Do_Not_Buy',
                                     'Buy_Insurance',
                                     ' '                );
array COST{4} $        _TEMPORARY_ ('Low',
                                     'Fair',
                                     'High',
                                     ' '                );
array DEPOSIT{4} $     _TEMPORARY_ ('Dry',
                                     'Wet',
                                     'Soaking',
                                     ' '                 );

do i=1 to 10;            /* loop for each divestment */
   _STATE1=DIVEST{i};

   /*
    * determine the percentage of ownership retained
    * for this scenario
    */
   PCT=1.0-((i-1)*0.1);

   do j=1 to 2;       /* loop for insurance decision */
      _STATE2=INSUR{j};

      /*
       * determine the premium need to pay for this
       * scenario
       */
      if _STATE2='Buy_Insurance' then PREMIUM=130000;
      else                            PREMIUM=0;

      do k=1 to 3;        /* loop for each well cost */
         _STATE3=COST{k};

         /* determine the cost for this scenario */
         if      _STATE3='Low' then  _COST_=150000;
         else if _STATE3='Fair' then _COST_=300000;
         else                        _COST_=500000;

         do l=1 to 3;  /* loop for each deposit type */
```

```
            _STATE4=DEPOSIT{1};

            /*
             * determine the oil deposit and the
             * corresponding net payoff for this scenario
             */
            if      _STATE4='Dry' then _PAYOFF_=0;
            else if _STATE4='Wet' then _PAYOFF_=700000;
            else                       _PAYOFF_=1200000;

            /* determine redeem received for this scenario */
            if _STATE2='Buy_Insurance' and _STATE4='Dry' then
                 REDEEM=200000;
            else REDEEM=0;

            /* calculate the net return for this scenario */
            _VALUE_=(_PAYOFF_-_COST_-PREMIUM+REDEEM)*PCT;

            /* drop unneeded variables */
            drop _COST_ _PAYOFF_ PREMIUM REDEEM PCT;

            /* output this record */
            output;
         end;
      end;
   end;
end;

/* output an observation for the scenario 100%_Divestment */
_STATE1=DIVEST{11};
_STATE2=INSUR{3};
_STATE3=COST{4};
_STATE4=DEPOSIT{4};
_VALUE_=0;
output;

run;
```

The Dtoilu4 data set for this problem, which contains 181 observations and 5 variables, is displayed in Output 7.2.1.

Output 7.2.1 Payoffs of the Oil Wildcatter's Problem with Risk Sharing

Oil Wildcatter's Problem

The Payoffs

Obs	_STATE1	_STATE2	_STATE3	_STATE4	_VALUE_
1	No_Divestment	Do_Not_Buy	Low	Dry	$-150,000
2	No_Divestment	Do_Not_Buy	Low	Wet	$550,000
3	No_Divestment	Do_Not_Buy	Low	Soaking	$1,050,000
4	No_Divestment	Do_Not_Buy	Fair	Dry	$-300,000
5	No_Divestment	Do_Not_Buy	Fair	Wet	$400,000
6	No_Divestment	Do_Not_Buy	Fair	Soaking	$900,000
7	No_Divestment	Do_Not_Buy	High	Dry	$-500,000
8	No_Divestment	Do_Not_Buy	High	Wet	$200,000
9	No_Divestment	Do_Not_Buy	High	Soaking	$700,000
10	No_Divestment	Buy_Insurance	Low	Dry	$-80,000
11	No_Divestment	Buy_Insurance	Low	Wet	$420,000
12	No_Divestment	Buy_Insurance	Low	Soaking	$920,000
13	No_Divestment	Buy_Insurance	Fair	Dry	$-230,000
14	No_Divestment	Buy_Insurance	Fair	Wet	$270,000
15	No_Divestment	Buy_Insurance	Fair	Soaking	$770,000
16	No_Divestment	Buy_Insurance	High	Dry	$-430,000
17	No_Divestment	Buy_Insurance	High	Wet	$70,000
18	No_Divestment	Buy_Insurance	High	Soaking	$570,000
19	10%_Divestment	Do_Not_Buy	Low	Dry	$-135,000
20	10%_Divestment	Do_Not_Buy	Low	Wet	$495,000
21	10%_Divestment	Do_Not_Buy	Low	Soaking	$945,000
22	10%_Divestment	Do_Not_Buy	Fair	Dry	$-270,000
23	10%_Divestment	Do_Not_Buy	Fair	Wet	$360,000
24	10%_Divestment	Do_Not_Buy	Fair	Soaking	$810,000
25	10%_Divestment	Do_Not_Buy	High	Dry	$-450,000
26	10%_Divestment	Do_Not_Buy	High	Wet	$180,000
27	10%_Divestment	Do_Not_Buy	High	Soaking	$630,000
28	10%_Divestment	Buy_Insurance	Low	Dry	$-72,000
29	10%_Divestment	Buy_Insurance	Low	Wet	$378,000
30	10%_Divestment	Buy_Insurance	Low	Soaking	$828,000
31	10%_Divestment	Buy_Insurance	Fair	Dry	$-207,000
32	10%_Divestment	Buy_Insurance	Fair	Wet	$243,000
33	10%_Divestment	Buy_Insurance	Fair	Soaking	$693,000
34	10%_Divestment	Buy_Insurance	High	Dry	$-387,000
35	10%_Divestment	Buy_Insurance	High	Wet	$63,000
36	10%_Divestment	Buy_Insurance	High	Soaking	$513,000
37	20%_Divestment	Do_Not_Buy	Low	Dry	$-120,000
38	20%_Divestment	Do_Not_Buy	Low	Wet	$440,000
39	20%_Divestment	Do_Not_Buy	Low	Soaking	$840,000
40	20%_Divestment	Do_Not_Buy	Fair	Dry	$-240,000
41	20%_Divestment	Do_Not_Buy	Fair	Wet	$320,000
42	20%_Divestment	Do_Not_Buy	Fair	Soaking	$720,000
43	20%_Divestment	Do_Not_Buy	High	Dry	$-400,000
44	20%_Divestment	Do_Not_Buy	High	Wet	$160,000
45	20%_Divestment	Do_Not_Buy	High	Soaking	$560,000

Output 7.2.1 *continued*

Oil Wildcatter's Problem

The Payoffs

Obs	_STATE1	_STATE2	_STATE3	_STATE4	_VALUE_
46	20%_Divestment	Buy_Insurance	Low	Dry	$-64,000
47	20%_Divestment	Buy_Insurance	Low	Wet	$336,000
48	20%_Divestment	Buy_Insurance	Low	Soaking	$736,000
49	20%_Divestment	Buy_Insurance	Fair	Dry	$-184,000
50	20%_Divestment	Buy_Insurance	Fair	Wet	$216,000
51	20%_Divestment	Buy_Insurance	Fair	Soaking	$616,000
52	20%_Divestment	Buy_Insurance	High	Dry	$-344,000
53	20%_Divestment	Buy_Insurance	High	Wet	$56,000
54	20%_Divestment	Buy_Insurance	High	Soaking	$456,000
55	30%_Divestment	Do_Not_Buy	Low	Dry	$-105,000
56	30%_Divestment	Do_Not_Buy	Low	Wet	$385,000
57	30%_Divestment	Do_Not_Buy	Low	Soaking	$735,000
58	30%_Divestment	Do_Not_Buy	Fair	Dry	$-210,000
59	30%_Divestment	Do_Not_Buy	Fair	Wet	$280,000
60	30%_Divestment	Do_Not_Buy	Fair	Soaking	$630,000
61	30%_Divestment	Do_Not_Buy	High	Dry	$-350,000
62	30%_Divestment	Do_Not_Buy	High	Wet	$140,000
63	30%_Divestment	Do_Not_Buy	High	Soaking	$490,000
64	30%_Divestment	Buy_Insurance	Low	Dry	$-56,000
65	30%_Divestment	Buy_Insurance	Low	Wet	$294,000
66	30%_Divestment	Buy_Insurance	Low	Soaking	$644,000
67	30%_Divestment	Buy_Insurance	Fair	Dry	$-161,000
68	30%_Divestment	Buy_Insurance	Fair	Wet	$189,000
69	30%_Divestment	Buy_Insurance	Fair	Soaking	$539,000
70	30%_Divestment	Buy_Insurance	High	Dry	$-301,000
71	30%_Divestment	Buy_Insurance	High	Wet	$49,000
72	30%_Divestment	Buy_Insurance	High	Soaking	$399,000
73	40%_Divestment	Do_Not_Buy	Low	Dry	$-90,000
74	40%_Divestment	Do_Not_Buy	Low	Wet	$330,000
75	40%_Divestment	Do_Not_Buy	Low	Soaking	$630,000
76	40%_Divestment	Do_Not_Buy	Fair	Dry	$-180,000
77	40%_Divestment	Do_Not_Buy	Fair	Wet	$240,000
78	40%_Divestment	Do_Not_Buy	Fair	Soaking	$540,000
79	40%_Divestment	Do_Not_Buy	High	Dry	$-300,000
80	40%_Divestment	Do_Not_Buy	High	Wet	$120,000
81	40%_Divestment	Do_Not_Buy	High	Soaking	$420,000
82	40%_Divestment	Buy_Insurance	Low	Dry	$-48,000
83	40%_Divestment	Buy_Insurance	Low	Wet	$252,000
84	40%_Divestment	Buy_Insurance	Low	Soaking	$552,000
85	40%_Divestment	Buy_Insurance	Fair	Dry	$-138,000
86	40%_Divestment	Buy_Insurance	Fair	Wet	$162,000
87	40%_Divestment	Buy_Insurance	Fair	Soaking	$462,000
88	40%_Divestment	Buy_Insurance	High	Dry	$-258,000
89	40%_Divestment	Buy_Insurance	High	Wet	$42,000
90	40%_Divestment	Buy_Insurance	High	Soaking	$342,000

Output 7.2.1 *continued*

Oil Wildcatter's Problem

The Payoffs

Obs	_STATE1	_STATE2	_STATE3	_STATE4	_VALUE_
91	50%_Divestment	Do_Not_Buy	Low	Dry	$-75,000
92	50%_Divestment	Do_Not_Buy	Low	Wet	$275,000
93	50%_Divestment	Do_Not_Buy	Low	Soaking	$525,000
94	50%_Divestment	Do_Not_Buy	Fair	Dry	$-150,000
95	50%_Divestment	Do_Not_Buy	Fair	Wet	$200,000
96	50%_Divestment	Do_Not_Buy	Fair	Soaking	$450,000
97	50%_Divestment	Do_Not_Buy	High	Dry	$-250,000
98	50%_Divestment	Do_Not_Buy	High	Wet	$100,000
99	50%_Divestment	Do_Not_Buy	High	Soaking	$350,000
100	50%_Divestment	Buy_Insurance	Low	Dry	$-40,000
101	50%_Divestment	Buy_Insurance	Low	Wet	$210,000
102	50%_Divestment	Buy_Insurance	Low	Soaking	$460,000
103	50%_Divestment	Buy_Insurance	Fair	Dry	$-115,000
104	50%_Divestment	Buy_Insurance	Fair	Wet	$135,000
105	50%_Divestment	Buy_Insurance	Fair	Soaking	$385,000
106	50%_Divestment	Buy_Insurance	High	Dry	$-215,000
107	50%_Divestment	Buy_Insurance	High	Wet	$35,000
108	50%_Divestment	Buy_Insurance	High	Soaking	$285,000
109	60%_Divestment	Do_Not_Buy	Low	Dry	$-60,000
110	60%_Divestment	Do_Not_Buy	Low	Wet	$220,000
111	60%_Divestment	Do_Not_Buy	Low	Soaking	$420,000
112	60%_Divestment	Do_Not_Buy	Fair	Dry	$-120,000
113	60%_Divestment	Do_Not_Buy	Fair	Wet	$160,000
114	60%_Divestment	Do_Not_Buy	Fair	Soaking	$360,000
115	60%_Divestment	Do_Not_Buy	High	Dry	$-200,000
116	60%_Divestment	Do_Not_Buy	High	Wet	$80,000
117	60%_Divestment	Do_Not_Buy	High	Soaking	$280,000
118	60%_Divestment	Buy_Insurance	Low	Dry	$-32,000
119	60%_Divestment	Buy_Insurance	Low	Wet	$168,000
120	60%_Divestment	Buy_Insurance	Low	Soaking	$368,000
121	60%_Divestment	Buy_Insurance	Fair	Dry	$-92,000
122	60%_Divestment	Buy_Insurance	Fair	Wet	$108,000
123	60%_Divestment	Buy_Insurance	Fair	Soaking	$308,000
124	60%_Divestment	Buy_Insurance	High	Dry	$-172,000
125	60%_Divestment	Buy_Insurance	High	Wet	$28,000
126	60%_Divestment	Buy_Insurance	High	Soaking	$228,000
127	70%_Divestment	Do_Not_Buy	Low	Dry	$-45,000
128	70%_Divestment	Do_Not_Buy	Low	Wet	$165,000
129	70%_Divestment	Do_Not_Buy	Low	Soaking	$315,000
130	70%_Divestment	Do_Not_Buy	Fair	Dry	$-90,000
131	70%_Divestment	Do_Not_Buy	Fair	Wet	$120,000
132	70%_Divestment	Do_Not_Buy	Fair	Soaking	$270,000
133	70%_Divestment	Do_Not_Buy	High	Dry	$-150,000
134	70%_Divestment	Do_Not_Buy	High	Wet	$60,000
135	70%_Divestment	Do_Not_Buy	High	Soaking	$210,000

Output 7.2.1 *continued*

Oil Wildcatter's Problem

The Payoffs

Obs	_STATE1	_STATE2	_STATE3	_STATE4	_VALUE_
136	70%_Divestment	Buy_Insurance	Low	Dry	$-24,000
137	70%_Divestment	Buy_Insurance	Low	Wet	$126,000
138	70%_Divestment	Buy_Insurance	Low	Soaking	$276,000
139	70%_Divestment	Buy_Insurance	Fair	Dry	$-69,000
140	70%_Divestment	Buy_Insurance	Fair	Wet	$81,000
141	70%_Divestment	Buy_Insurance	Fair	Soaking	$231,000
142	70%_Divestment	Buy_Insurance	High	Dry	$-129,000
143	70%_Divestment	Buy_Insurance	High	Wet	$21,000
144	70%_Divestment	Buy_Insurance	High	Soaking	$171,000
145	80%_Divestment	Do_Not_Buy	Low	Dry	$-30,000
146	80%_Divestment	Do_Not_Buy	Low	Wet	$110,000
147	80%_Divestment	Do_Not_Buy	Low	Soaking	$210,000
148	80%_Divestment	Do_Not_Buy	Fair	Dry	$-60,000
149	80%_Divestment	Do_Not_Buy	Fair	Wet	$80,000
150	80%_Divestment	Do_Not_Buy	Fair	Soaking	$180,000
151	80%_Divestment	Do_Not_Buy	High	Dry	$-100,000
152	80%_Divestment	Do_Not_Buy	High	Wet	$40,000
153	80%_Divestment	Do_Not_Buy	High	Soaking	$140,000
154	80%_Divestment	Buy_Insurance	Low	Dry	$-16,000
155	80%_Divestment	Buy_Insurance	Low	Wet	$84,000
156	80%_Divestment	Buy_Insurance	Low	Soaking	$184,000
157	80%_Divestment	Buy_Insurance	Fair	Dry	$-46,000
158	80%_Divestment	Buy_Insurance	Fair	Wet	$54,000
159	80%_Divestment	Buy_Insurance	Fair	Soaking	$154,000
160	80%_Divestment	Buy_Insurance	High	Dry	$-86,000
161	80%_Divestment	Buy_Insurance	High	Wet	$14,000
162	80%_Divestment	Buy_Insurance	High	Soaking	$114,000
163	90%_Divestment	Do_Not_Buy	Low	Dry	$-15,000
164	90%_Divestment	Do_Not_Buy	Low	Wet	$55,000
165	90%_Divestment	Do_Not_Buy	Low	Soaking	$105,000
166	90%_Divestment	Do_Not_Buy	Fair	Dry	$-30,000
167	90%_Divestment	Do_Not_Buy	Fair	Wet	$40,000
168	90%_Divestment	Do_Not_Buy	Fair	Soaking	$90,000
169	90%_Divestment	Do_Not_Buy	High	Dry	$-50,000
170	90%_Divestment	Do_Not_Buy	High	Wet	$20,000
171	90%_Divestment	Do_Not_Buy	High	Soaking	$70,000
172	90%_Divestment	Buy_Insurance	Low	Dry	$-8,000
173	90%_Divestment	Buy_Insurance	Low	Wet	$42,000
174	90%_Divestment	Buy_Insurance	Low	Soaking	$92,000
175	90%_Divestment	Buy_Insurance	Fair	Dry	$-23,000
176	90%_Divestment	Buy_Insurance	Fair	Wet	$27,000
177	90%_Divestment	Buy_Insurance	Fair	Soaking	$77,000
178	90%_Divestment	Buy_Insurance	High	Dry	$-43,000
179	90%_Divestment	Buy_Insurance	High	Wet	$7,000

Output 7.2.1 *continued*

Oil Wildcatter's Problem

The Payoffs

Obs	_STATE1	_STATE2	_STATE3	_STATE4	_VALUE_
180	90%_Divestment	Buy_Insurance	High	Soaking	$57,000
181	100%_Divestment				$0

The optimal decisions for this problem can be identified by invoking PROC DTREE and using the SUMMARY statement as follows:

```
title "Oil Wildcatter's Problem";
proc dtree stagein=Dtoils4
           probin=Dtoilp4
           payoffs=Dtoilu4
           criterion=maxce rt=1200000
           nowarning;
   evaluate;
   summary / target=Divestment;
   summary / target=Insurance;
quit;
```

The optimal decision summaries in Output 7.2.2 and Output 7.2.3 show the optimal strategy for the wildcatter.

- The wildcatter should sell 30% of his investment to other companies and reject the insurance policy offered to him.
- The insurance policy should be accepted only if the decision to not divest is made.
- If the decision to buy the insurance policy is made, then it is optimal to divest 10% of the venture.

Output 7.2.2 Summary of the Oil Wildcatter's Problem for DIVESTMENT

Oil Wildcatter's Problem

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Divestment	Decision
Insurance	Decision
Cost	Chance
Oil_Deposit	Chance
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Certain Equivalent Value (MAXCE)
Risk Tolerance:	$1,200,000
Optimal Decision Yields:	$50,104

Output 7.2.2 *continued*

Optimal Decision Policy Up to Stage Divestment		
Alternatives or Outcomes	**Cumulative Reward**	**Evaluating Value**
No_Divestment		$45,728
10%_Divestment		$48,021
20%_Divestment		$49,907
30%_Divestment		$50,104*
40%_Divestment		$48,558
50%_Divestment		$45,219
60%_Divestment		$40,036
70%_Divestment		$32,965
80%_Divestment		$23,961
90%_Divestment		$12,985
100%_Divestment		$0

Output 7.2.3 Summary of the Oil Wildcatter's Problem for INSURANCE

Oil Wildcatter's Problem

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Divestment	Decision
Insurance	Decision
Cost	Chance
Oil_Deposit	Chance
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Certain Equivalent Value (MAXCE)
Risk Tolerance:	$1,200,000
Optimal Decision Yields:	$50,104

Output 7.2.3 *continued*

Optimal Decision Policy Up to Stage Insurance			
Alternatives or Outcomes		Cumulative Reward	Evaluating Value
No_Divestment	Buy_Insurance		$45,728*
No_Divestment	Do_Not_Buy		$44,499
10%_Divestment	Buy_Insurance		$46,552
10%_Divestment	Do_Not_Buy		$48,021*
20%_Divestment	Buy_Insurance		$46,257
20%_Divestment	Do_Not_Buy		$49,907*
30%_Divestment	Buy_Insurance		$44,812
30%_Divestment	Do_Not_Buy		$50,104*
40%_Divestment	Buy_Insurance		$42,186
40%_Divestment	Do_Not_Buy		$48,558*
50%_Divestment	Buy_Insurance		$38,350
50%_Divestment	Do_Not_Buy		$45,219*
60%_Divestment	Buy_Insurance		$33,273
60%_Divestment	Do_Not_Buy		$40,036*
70%_Divestment	Buy_Insurance		$26,927
70%_Divestment	Do_Not_Buy		$32,965*
80%_Divestment	Buy_Insurance		$19,284
80%_Divestment	Do_Not_Buy		$23,961*
90%_Divestment	Buy_Insurance		$10,317
90%_Divestment	Do_Not_Buy		$12,985*

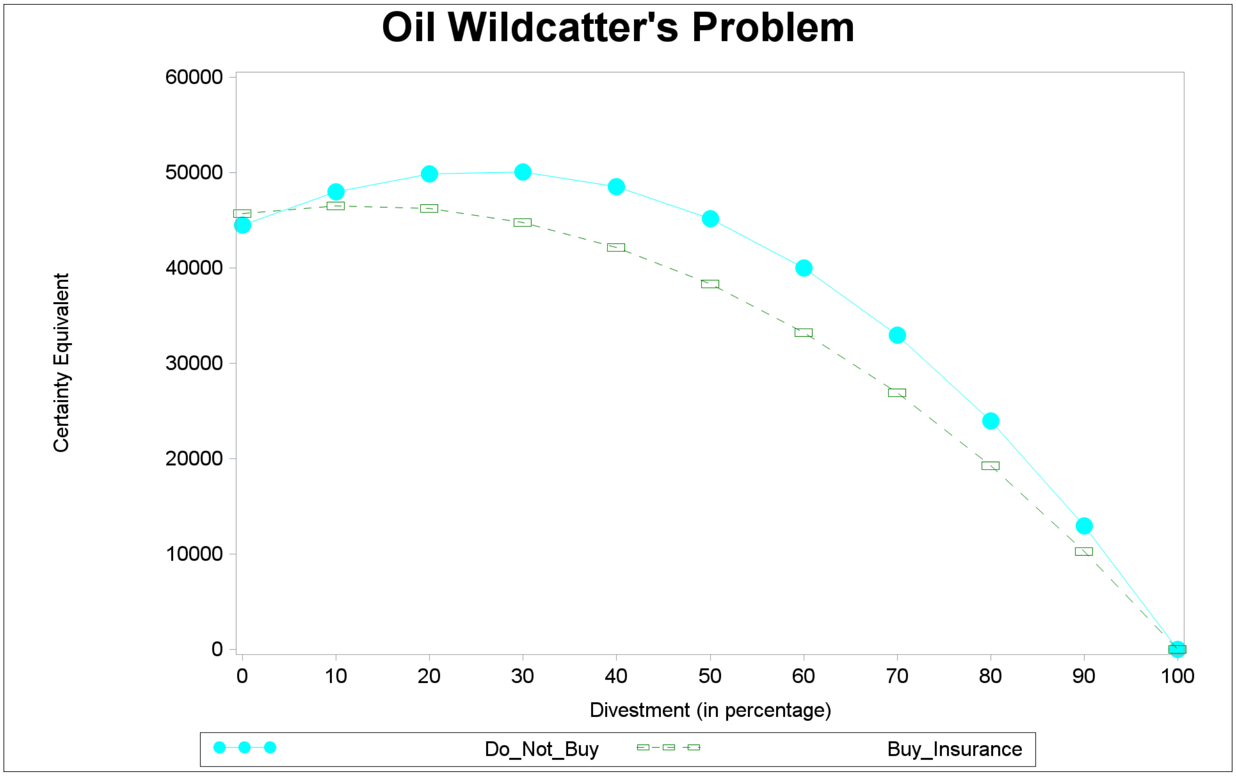
This information can be illustrated graphically using the GPLOT procedure. Output 7.2.4, produced by the PROC GPLOT statements shown in the following code, provides a clear picture of the effects of the divestment possibilities and the insurance options.

```
    /*  create a data set for the return corresponds to each */
    /*  divestment possibilities and the insurance options   */
 data Data2g;
    input  INSURE DIVEST VALUE;
    datalines;
        1       0   45728
        0       0   44499
        1      10   46552
        0      10   48021
        1      20   46257
        0      20   49907
        1      30   44812
        0      30   50104
        1      40   42186
        0      40   48558
        1      50   38350
        0      50   45219
        1      60   33273
        0      60   40036
        1      70   26927
        0      70   32965
```

```
       1      80   19284
       0      80   23961
       1      90   10317
       0      90   12985
       1     100       0
       0     100       0
;

   /* -- define a format for INSURE variable             -- */
proc format;
   value sample 0='Do_Not_Buy' 1='Buy_Insurance';
run;

   /* -- define title                                    -- */
title h=3 "Oil Wildcatter's Problem";

   /* define legend                                      -- */
legend1 frame cframe=white label=none
        cborder=black position=center ;

   /* define symbol characteristics of the data points      */
   /* and the interpolation line for returns vs divestment  */
   /* when INSURE=0                                          */
symbol1 c=cyan i=join v=dot l=1 h=1;

   /* define symbol characteristics of the data points      */
   /* and the interpolation line for returns vs divestment  */
   /*  when INSURE=1                                         */
symbol2 c=green i=join v=square l=2 h=1;

   /* -- define axis characteristics                     -- */
axis1 minor=none label=('Divestment (in percentage)');
axis2 minor=none label=(angle=90 rotate=0 'Certainty Equivalent');

   /* set graphics options                                   */
goptions htext=1.5;

   /* plot VALUE vs DIVEST using INSURE as third variable   */
proc gplot data=Data2g ;
   plot VALUE*DIVEST=INSURE / haxis=axis1
                              vaxis=axis2
                              legend=legend1
                              name="dt2"
                              frame
                              cframe=white ;
   format INSURE SAMPLE.;
run;

quit;
```

Note that the data input into the Data2g data set is obtained from the optimal decision summary as in Output 7.2.3. The value 1 of the INSURE variable represents the alternative **`'Buy_Insurance'`** and the value 0 represents the alternative **`'Do_Not_Buy'`**.

Output 7.2.4 Returns of the Oil Wildcatter's Problem



Example 7.3: Contract Bidding Problem

This example illustrates the use of several of the graphics options for producing graphics quality decision tree diagrams.

The production manager of a manufacturing company is planning to bid on a project to manufacture a new type of machine. He has the choice of bidding low or high. The evaluation of the bid will more likely be favorable if the bidder has built a prototype of the machine and includes it with the bid. However, he is uncertain about the cost of building the prototype.- His technical staff has provided him a probability distribution on the cost of the prototype.

Table 7.20 Probability on the Cost of Building Prototype

Outcome	Cost	Probability
Expensive	$4,500	0.4
Moderate	$2,500	0.5
Inexpensive	$1,000	0.1

There is also uncertainty in whether he will win the contract or not. He has estimated the probability distribution of winning the contract as shown in Table 7.21.

Table 7.21 Probability of Winning the Contract

		Events	
Givens		Win the Contract	Lose the Contract
Build Prototype	High Bid	0.4	0.6
Build Prototype	Low Bid	0.8	0.2
No Prototype	High Bid	0.2	0.8
No Prototype	Low Bid	0.7	0.3

In addition, the payoffs of this bidding venture are affected by the cost of building the prototype. Table 7.22 shows his payoffs. The first row of the table shows the payoff is 0 if he loses the contract, regardless of whether or not he builds the prototype and whether he bids low or high. The remainder of the entries in the table give the payoff under the various scenarios.

Table 7.22 Payoffs of the Contract Bidding Decision

States		**Actions**	
Result	Cost	Bid low	Bid high
Lose the Contract		0	0
Win the Contract		$35,000	$75,000
Win the Contract	Expensive	$25,000	$65,000
Win the Contract	Moderate	$35,000	$75,000
Win the Contract	Inexpensive	$45,000	$85,000

The production manager must decide whether to build the prototype and how to bid. He uses PROC DTREE to help him to make these decisions. The structure of the model is stored in the STAGEIN= data set named Stage3. There are two decision stages, `'Choose'` and `'Bid'`, and two chance stages, `'Cost_Prototype'` and `'Contract'`. The `'Choose'` stage represents the decision whether or not to build a prototype. The chance stage `'Cost_Prototype'` represents the uncertain cost for building a prototype. It can be `'Expensive'`, which costs $4,500, or `'Moderate'`, which costs $2,500, or `'Inexpensive'`, which costs $1,000. The `'Bid'` stage represents the decision whether to bid high or bid low. The last stage, `'Contract'`, represents the result, either win the contract or lose the contract.

```
/* -- create the STAGEIN= data set                    -- */
data Stage3;
format _STNAME_ $14. _STTYPE_ $2. _OUTCOM_ $15.
     _SUCCES_ $14. _REWARD_ dollar8.0 ;
input _STNAME_ $16. _STTYPE_ $4. _OUTCOM_ $16.
    _SUCCES_ $16. _REWARD_ dollar8.0 ;
datalines;
Choose          D   Build_Prototype Cost_Prototype      .
.               .   No_Prototype    Bid                 .
Cost_Prototype  C   Expensive       Bid            -$4,500
.               .   Moderate        Bid            -$2,500
.               .   Inexpensive     Bid            -$1,000
Bid             D   High_Bid        Contract            .
.               .   Low_Bid         Contract            .
Contract        C   Win_Contract    .                   .
.               .   Lose_Contract   .                   .
;
```

The PROBIN= data set, named Prob3, contains the probability information as in Table 7.20 and Table 7.21.

```
/* -- create the PROBIN= data set                        -- */
data Prob3;
format _GIVEN1_ $15. _GIVEN2_ $15. _EVENT_ $14. ;
input (_GIVEN1_ _GIVEN2_ _EVENT_) ($) _PROB_;
datalines;
.               .               Expensive       0.4
.               .               Moderate        0.5
.               .               Inexpensive     0.1
Build_Prototype High_Bid        Win_Contract    0.4
Build_Prototype High_Bid        Lose_Contract   0.6
Build_Prototype Low_Bid         Win_Contract    0.8
Build_Prototype Low_Bid         Lose_Contract   0.2
No_Prototype    High_Bid        Win_Contract    0.2
No_Prototype    High_Bid        Lose_Contract   0.8
No_Prototype    Low_Bid         Win_Contract    0.7
No_Prototype    Low_Bid         Lose_Contract   0.3
;
```

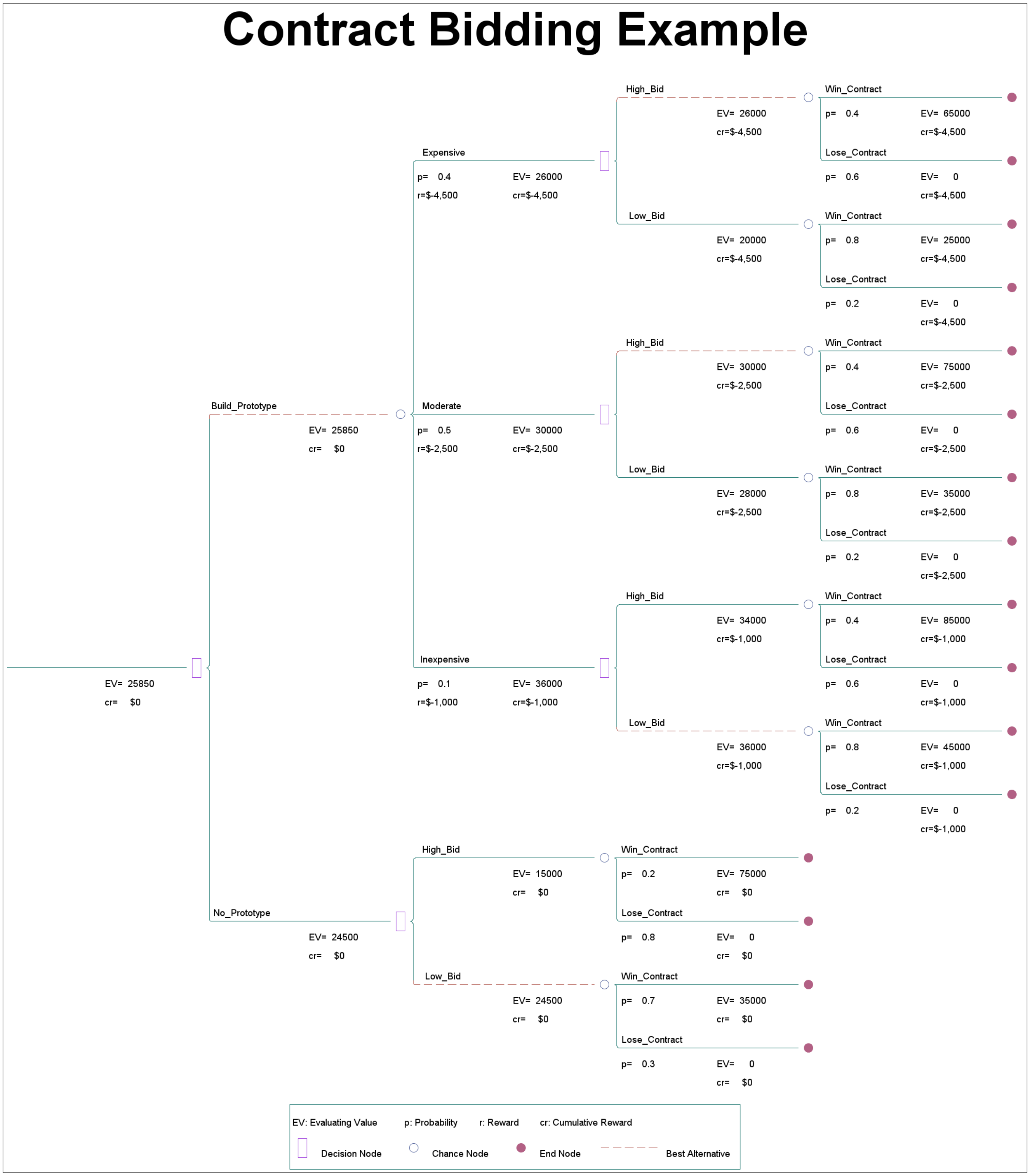
The PAYOFFS= data set named Payoff3 contains the payoff information as in Table 7.22. Notice that the payoff to outcome `'Lose_Contract'` is not in the data set Payoff3. Since PROC DTREE assigns the default value 0 to all scenarios that are not in the PAYOFFS= data set, it is not necessary to include it.

```
/* -- create the PAYOFFS= data set                      -- */
data Payoff3;
format _STATE1_ _STATE2_ $12.;
input (_STATE1_ _STATE2_ _ACTION_) ($16.)
    _VALUE_  dollar8.0;
datalines;
Win_Contract    .               Low_Bid         $35,000
Win_Contract    .               High_Bid        $75,000
Win_Contract    Expensive       Low_Bid         $25,000
Win_Contract    Expensive       High_Bid        $65,000
Win_Contract    Moderate        Low_Bid         $35,000
Win_Contract    Moderate        High_Bid        $75,000
Win_Contract    Inexpensive     Low_Bid         $45,000
Win_Contract    Inexpensive     High_Bid        $85,000
;
```

The solution, as in Output 7.3.1, is displayed on a graphics device with the following code. Notice that the title is specified before invoking PROC DTREE. The GRAPHICS option is given on the PROC DTREE statement. Specifying the COMPRESS option in the TREEPLOT statement causes the decision tree diagram to be drawn completely on one page. The vertical distance between two successive end nodes is 1 character cell (**ybetween=1 cell**). All text, except that in the first title line, is drawn with the font specified by the FTEXT= option.The height for all nodes is the number of character cells specified by the HSYMBOL= option. The thickness for all links in the diagram, except those that represent optimal decisions, is specified by the LWIDTH= option. The thickness of the links that represent optimal decisions is specified by the LWIDTHB= option, and the type of those links is 3 (**lstyleb=3**), the dash line. Colors for the text, links and nodes, and symbols to be used for nodes are not specified and hence defaults are used.

```
goptions ctext=black;
goptions hsize=10in htext=3.0;
 /* -- define title                                   -- */
title1 h=2 "Contract Bidding Example" ;

 /* -- PROC DTREE statements                          -- */
proc dtree stagein=Stage3 probin=Prob3 payoffs=Payoff3
     graphics
     nowarning
     ;
  evaluate;
  treeplot / name="dt3" compress ybetween=1 cell
             hsymbol=6
             lstyleb=3 lwidth=1 lwidthb=1;
quit;
```

Output 7.3.1 Decision Tree for the Contract Bidding Problem



With the information on this decision tree, the production manager can select the optimal bidding strategy:

- He should build a prototype to accompany the bid and always bid high unless the cost for building the prototype is as low as $1,000. This optimal strategy yields an expected return of $25,850.
- If no prototype is built, the preferred decision is to make a low bid. In this case the expected return is $24,500.

Example 7.4: Research and Development Decision Problem

This example illustrates the use of the SYMBOL and GOPTIONS statements for controlling the appearance of the decision tree diagram. It also uses the ANNOTATE= option to add a customized legend to the diagram.

A typical problem encountered in a research and development setting involves two decisions: whether or not to conduct research, and whether or not to commercialize the results of that research. Suppose that research and development for a specific project will cost $350,000, and there is a 0.4 probability that it will fail. Also suppose that the different levels of market success and their corresponding probabilities are:

Table 7.23 Levels of Market Success and Their Probabilities

Market Success	Net Return	Probability
Great	$1,000,000	0.25
Good	$500,000	0.35
Fair	$200,000	0.30
Poor	-$250,000	0.10

The structure of the model is represented in the STAGEIN= data set Stage4.

```
/* -- create the STAGEIN= data set             -- */
data Stage4;
input _STNAME_ $ 1-16 _STTYPE_ $ 17-20
 _OUTCOM_ $ 21-32 _REWARD_ dollar12.0
 _SUCC_ $ 45-60;
datalines;
R_and_D         D   Not_Conduct .           .
.               .   Conduct     -$350,000   RD_Outcome
RD_Outcome      C   Success     .           Production
.               .   Failure     .           .
Production      D   Produce     .           Sales
.               .   Abandon     .           .
Sales           C   Great       .           .
.               .   Good        .           .
.               .   Fair        .           .
.               .   Poor        .           .
;
```

The probability distributions for the various outcomes of the chance stages are given in the PROBIN= data set named Prob4.

```
/* -- create the PROBIN= data set               -- */
data Prob4;
input _EVENT1_ $ _PROB1_ _EVENT2_ $12. _PROB2_;
datalines;
Success     0.6     Failure   0.4
Great       0.25    Good      0.35
Fair        0.30    poor      0.1
;
```

The payoffs are given in the PAYOFFS= data set Payoff4.

```
/* -- create the PAYOFFS= data set                     -- */
data Payoff4;
input _STATE_ $12. _VALUE_ dollar12.0;
datalines;
Great       $1,000,000
Good        $500,000
Fair        $200,000
Poor        -$250,000
;
```

The following DATA step builds a data set that contains the Annotate description of a legend. Refer to the chapter on the annotate facility in *SAS/GRAPH Software: Reference* for a description of the Annotate facility.

```
/* -- create the ANNOTATE= data set for legend     -- */
data Legend;
length FUNCTION $ 8;
length STYLE $ 16;
WHEN = 'B';  POSITION='0';
XSYS='4';  YSYS='4';
input FUNCTION $ X Y STYLE & 16. SIZE COLOR $ TEXT $ & 16.;
datalines;
move        8   2.1  .                .   .        .
draw       12   2.1  .                8   red      .
label      14   2    Cumberland AMT   0.6 black    BEST ACTION
symbol      9   3.5  marker           0.6 red      A
label      14   3.2  Cumberland AMT   0.6 black    END NODE
symbol      9   4.7  marker           0.6 blue     P
label      14   4.4  Cumberland AMT   0.6 black    CHANCE NODE
symbol      9   5.9  marker           0.6 green    U
label      14   5.6  Cumberland AMT   0.6 black    DECISION NODE
label       8   7.0  Cumberland AMT   0.6 black    LEGEND:
move        5   8.5  .                .   black    .
draw       35   8.5  .                2   black    .
draw       35   1    .                2   black    .
draw        5   1    .                2   black    .
draw        5   8.5  .                2   black    .
;
```

The following program invokes PROC DTREE, which evaluates the decision tree and plots it on a graphics device using the Annotate data set Legend to draw the legend.

```
   /* define symbol characteristics for chance nodes and  */
   /* links except those that represent optimal decisions */
symbol1 f=marker h=1.8 v=P c=blue w=5 l=1;

   /* define symbol characteristics for decision nodes    */
   /* and links that represent optimal decisions          */
symbol2 f=marker h=1.8 v=U cv=green ci=red w=10 l=1;

   /* define symbol characteristics for end nodes         */
symbol3 f=marker h=1.8 v=A cv=red;

   /* define graphics options                             */
goptions htext=1.2;

   /* -- define title -- */
title f='Cumberland AMT'
h=2.5 'Research and Development Decision';

   /* -- PROC DTREE statements                          -- */
proc dtree
     stagein=Stage4 probin=Prob4 payoffs=Payoff4
     criterion=maxce rt=1800000
     graphics annotate=Legend nolg ;

   evaluate;

   treeplot / linka=1 linkb=2
              symbold=2 symbolc=1 symbole=3 compress name="dt4";

quit;
```

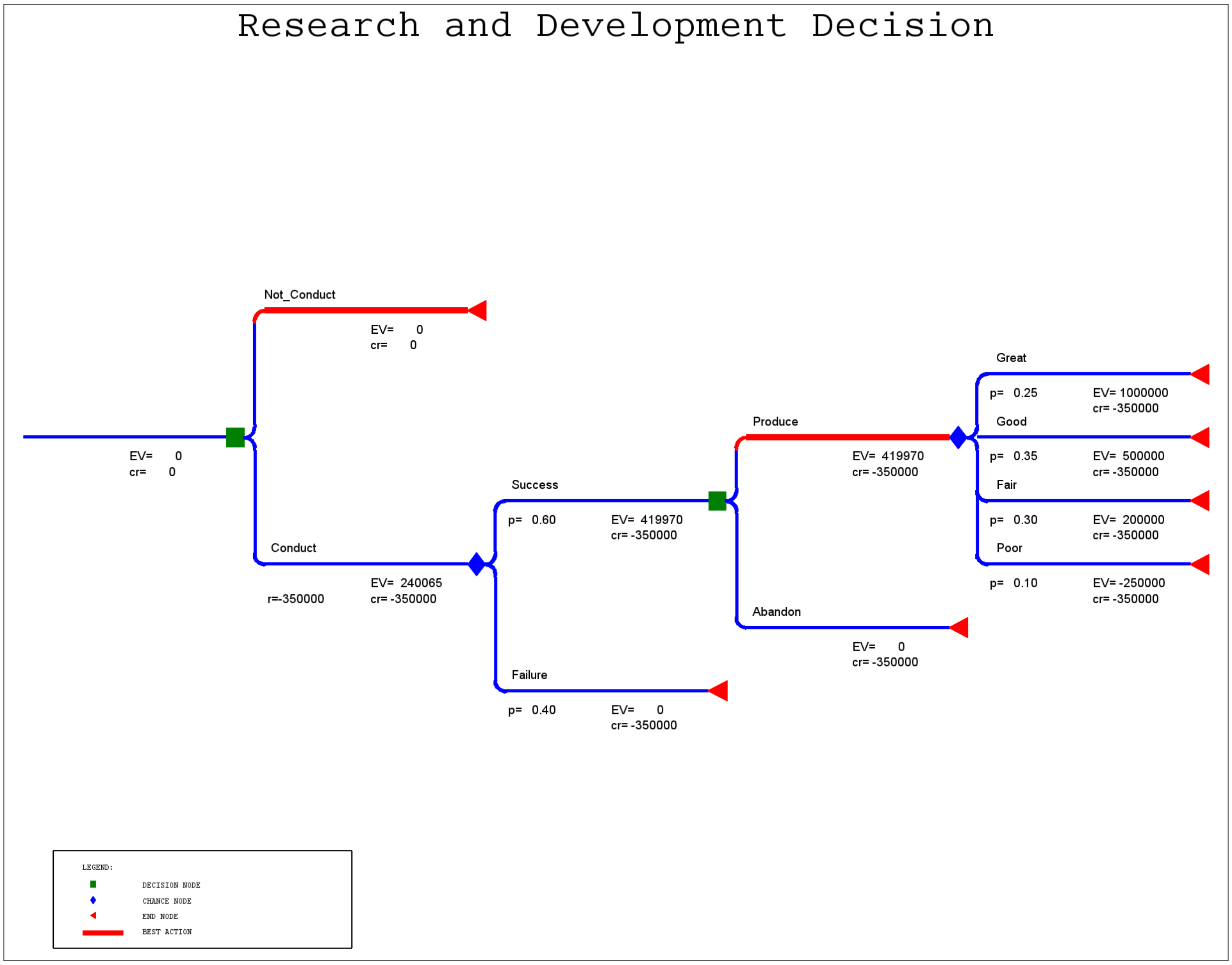
The SYMBOL1, SYMBOL2, and SYMBOL3 statements create three SYMBOL definitions that contain information for drawing nodes and links. The Legend data set and the ANNOTATE= option specified in the PROC DTREE statement cause the procedure to produce a customized legend for the decision tree diagram. The LINKA=, LINKB=, SYMBOLD=, SYMBOLC=, and SYMBOLE= specifications in the TREEPLOT statement tell PROC DTREE how to use SYMBOL definitions to draw the decision tree. Table 7.24 describes the options in SYMBOL definitions used to draw the decision tree diagram.

The decision tree diagram produced by the TREEPLOT statement is shown in Output 7.4.1. As illustrated on the decision tree, the program recommends that one should not conduct the research and development of the product if he or she is risk averse with a risk tolerance of $1,800,000. However, if he or she decides to undertake the research and development and it is a success, then he or she should commercialize the product.

Table 7.24 The Usage of SYMBOL Definitions

SYMBOL Definition	Specification	Description	Used to Draw
The First	C=*blue*	Color	All links except
	L=1	Line Type	those that indicate
	W=3	Thickness	optimal decisions
	C=*blue*	Color	Chance nodes
	F=*marker*	Font	
	H=2	Height	
	V=*P*	Symbol	
The Second	CI=*red*	Color	All links that
	L=1	Line Type	indicate optimal
	W=3	Thickness	decisions
	CV=*green*	Color	Decision nodes
	F=*marker*	Font	
	H=2	Height	
	V=*U*	Symbol	
The Third	CV=*red*	Color	End nodes
	F=*marker*	Font	
	H=2	Height	
	V=*A*	Symbol	

Output 7.4.1 Research and Development Decision Tree



Example 7.5: Loan Grant Decision Problem

Many financial decisions are difficult to analyze because of the variety of available strategies and the continuous nature of the problem. However, if the alternatives and time frame can be restricted, then decision analysis can be a useful analysis tool.

For example, a loan officer is faced with the problem of deciding whether to *approve* or *deny* an application for a one-year $30,000 loan at the current rate of 15% of interest. If the application is approved, the borrower will either *pay off* the loan in full after one year or *default*. Based on experience, the default rate is about 36 out of 700. If the loan is denied, the money is put in government bonds at the interest rate of 8%.

To obtain more information about the applicant, the loan officer engages a credit investigation unit at a cost of $500 per person that will give either a *positive* recommendation for making a loan or a *negative* recommendation. Past experience with this investigator yields that of those who ultimately paid off their loans, 570 out of 664 were given a positive recommendation. On the other hand, 6 out of 26 that had defaulted had also been given a positive recommendation by the investigator.

The STAGEIN= data set, Stage6, gives the structure of the decision problem.

```
/* -- create the STAGEIN= data set                    -- */
data Stage6;
format _STNAME_ $14. _STTYPE_ $2. _OUTCOM_ $20. _SUCC_ $14. ;
input _STNAME_ $ _STTYPE_ $ _OUTCOM_ & _SUCC_ $ ;
datalines;
Application     D   Approve loan            Payment
.               .   Deny loan               .
Payment         C   Pay off                 .
.               .   Default                 .
Investigation   D   Order investigation     Recommendation
.               .   Do not order            Application
Recommendation  C   Positive                Application
.               .   Negative                Application
;
```

The PROBIN= data set Prob6 gives the probability distributions for the random events at the chance nodes.

```
/* -- create the PROBIN= data set                     -- */
data Prob6;
length _GIVEN_ _EVENT1_ _EVENT2_ $16;

_EVENT1_='Pay off';    _EVENT2_='Default';
_PROB1_=664/700;       _PROB2_=1.0-_PROB1_;
output;

_GIVEN_='Pay off';
_EVENT1_='Positive';   _EVENT2_='Negative';
_PROB1_=570/664;       _PROB2_=1.0-_PROB1_;
output;

_GIVEN_='Default';
_EVENT1_='Positive';   _EVENT2_='Negative';
_PROB1_=6/26;          _PROB2_=1.0-_PROB1_;
output;

run;
```

The PAYOFFS= data set Payoff6 gives the payoffs for the various scenarios. Notice that the first observation in this data set indicates that if the officer denies the loan application, then payoffs are the interest from the money invested in government bonds. The second and the third observations are redundant for the basic analysis but are needed to determine the value of information as shown later.

```
/* -- create the PAYOFFS= data set                    -- */
data Payoff6(drop=loan);
length _STATE_ _ACT_ $24;

loan=30000;

_ACT_='Deny loan';    _VALUE_=loan*0.08;   output;
_STATE_='Pay off';    _VALUE_=loan*0.08;   output;
_STATE_='Default';    _VALUE_=loan*0.08;   output;

_ACT_='Approve loan';
```

```
_STATE_='Pay off';   _VALUE_=loan*0.15;   output;
_STATE_='Default';   _VALUE_=-1.0*loan;   output;

run;
```

The following code invokes the DTREE procedure to solve this decision problem.

```
   /* -- define title                                    -- */
title 'Loan Grant Decision';

   /* -- PROC DTREE statements                           -- */
proc dtree
     stagein=Stage6 probin=Prob6 payoffs=Payoff6
     summary target=investigation nowarning;

   modify 'Order investigation' reward -500;

   evaluate;

   OPTIONS LINESIZE=85;
   summary / target=Application;
   OPTIONS LINESIZE=80;
```

Note that the $500 investigation fee is not included in the Stage6 data set. Since the outcome `'Order investigation'` is the only outcome that has a nonzero reward, it is easier to set the reward for this outcome using the MODIFY statement. The quotes that enclose the outcome name in the MODIFY statement are necessary because the outcome name contains a space.

The results in Output 7.5.1 and Output 7.5.2 indicate that it is optimal to do the following:

- The loan officer should order the credit investigation and approve the loan application if the investigator gives the applicant a positive recommendation. On the other hand, he should deny the application if a negative recommendation is given to the applicant.
- Furthermore, the loan officer should order a credit investigation if the cost for the investigation is less than $\$3,725 - \$2,726 = \$999$.

Output 7.5.1 Summary of the Loan Grant Decision for Investigation

Loan Grant Decision

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Investigation	Decision
Recommendation	Chance
Application	Decision
Payment	Chance
ENDST	End

Output 7.5.1 *continued*

Decision Parameters	
Decision Criterion:	Maximize Expected Value (MAXEV)
Optimal Decision Yields:	3225

Optimal Decision Policy
Up to Stage Investigation

Alternatives or Outcomes	Cumulative Reward	Evaluating Value
Order investigation	-500	3725*
Do not order	0	2726

Output 7.5.2 Summary of the Loan Grant Decision for Application

Loan Grant Decision

The DTREE Procedure

Optimal Decision Summary

Order of Stages

Stage	Type
Investigation	Decision
Recommendation	Chance
Application	Decision
Payment	Chance
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Expected Value (MAXEV)
Optimal Decision Yields:	3225

Optimal Decision Policy
Up to Stage Application

Alternatives or Outcomes			Cumulative Reward	Evaluating Value
Order investigation	**Positive**	**Approve loan**	-500	4004*
Order investigation	**Positive**	**Deny loan**	-500	2400
Order investigation	**Negative**	**Approve loan**	-500	-3351
Order investigation	**Negative**	**Deny loan**	-500	2400*
Do not order		**Approve loan**	0	2726*
Do not order		**Deny loan**	0	2400

Now, the loan officer learns of another credit investigation company that claims to have a more accurate credit checking system for predicting whether the applicants will default on their loans. However, he has not been able to find out what the company charges for their service or how accurate their credit checking system is. Perhaps the best thing he can do at this stage is to assume that the company can predict perfectly whether or not applicants will default on their loans and determine the maximum amount to pay for this perfect investigation. The answer to this question can be found with the PROC DTREE statements:

```
save;
move payment before investigation;
evaluate;
recall;
```

Notice that moving the stage `'Payment'` to the beginning of the tree means that the new decision tree contains two scenarios that are not in the original tree: the scenario `'Pay off'` and `'Deny loan'`, and the scenario `'Default'` and `'Deny loan'`. The second and third observations in the Payoff6 data set supply values for these new scenarios. If these records are not included in the PAYOFFS= data set, then PROC DTREE assumes they are 0.

Also notice that the SUMMARY and TARGET= options are specified globally in the PROC DTREE statement and hence are not needed in the EVALUATE statement. The results from the DTREE procedure are displayed in Output 7.5.3.

Output 7.5.3 Summary of the Loan Grant Decision with Perfect Information

Loan Grant Decision

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Payment	Chance
Investigation	Decision
Recommendation	Chance
Application	Decision
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Expected Value (MAXEV)
Optimal Decision Yields:	4392

Optimal Decision Policy
Up to Stage Investigation

Alternatives or Outcomes		Cumulative Reward	Evaluating Value
Pay off	**Order investigation**	-500	4500
Pay off	**Do not order**	0	4500*
Default	**Order investigation**	-500	2400
Default	**Do not order**	0	2400*

The optimal decision summary in Output 7.5.3 shows that the yields with perfect investigation is $4,392. Recall that the yield of alternative `'Do not order'` the investigation, as shown in Output 7.5.1, is $2,726.

Therefore, the maximum amount he should pay for the perfect investigation can be determined easily as

$$\begin{aligned}\text{VPI} &= \text{Value with Perfect Investigation} - \text{Value without Investigation} \\ &= \$4,392 - \$2,726 \\ &= \$1,666\end{aligned}$$

Note that if you use the VPI statement to determine the value of a perfect investigation, the result is different from the value calculated previously.

```
vpi payment;
```

```
NOTE: The currently optimal decision yields 3225.4725275.
NOTE: The new optimal decision yields 4392.
NOTE: The value of perfect information of stage Payment
      yields 1166.5274725.
```

The reason for this difference is that the VPI statement causes PROC DTREE first to determine the value with perfect information, then to compare this value with the value with current information available (in this example, it is the recommendation from the original investigation unit). Therefore, the VPI statement returns a value that is calculated as

$$\begin{aligned}\text{VPI} &= \text{Value with Perfect Information} - \text{Value with Current Information} \\ &= \$4,392 - \$3,225 \\ &= \$1,167\end{aligned}$$

The loan officer considered another question regarding the maximum amount he should pay to a company to help collect the principal and the interest if an applicant defaults on the loan. This question is similar to the question concerning the improvement that can be expected if he can control whether or not an applicant will default on his loan (of course he will always want the applicant to pay off in full after one year). The answer to this question can be obtained with the following statements:

```
modify payment type;
evaluate;
```

Output 7.5.4 Summary of the Loan Grant Decision with Perfect Control

Loan Grant Decision

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Investigation	Decision
Recommendation	Chance
Application	Decision
Payment	Decision
ENDST	End

Decision Parameters	
Decision Criterion:	Maximize Expected Value (MAXEV)
Optimal Decision Yields:	4500

Optimal Decision Policy Up to Stage Investigation		
Alternatives or Outcomes	**Cumulative Reward**	**Evaluating Value**
Order investigation	-500	4500
Do not order	0	4500*

The result is obvious and is shown in Output 7.5.4. Using a calculation similar to the one used to calculate the value of a perfect investigation, the maximum amount one should pay for this kind of service is $\$4,500 - \$2,726 = \$1,774$. As previously described, this value is different from the value obtained by using the VPC statement. In fact, if you specify the statement

```
vpc payment;
```

you get the value of VPC, which is $1,274.53, from the SAS log as

```
NOTE: The currently optimal decision yields 3225.4725275.
NOTE: The new optimal decision yields 4500.
NOTE: The value of perfect control of stage Payment yields
      1274.5274725.
```

Obviously, all of the values of investigation and other services depend on the value of the loan. Since each of the payoffs for the various scenarios given in the Payoff6 data set is proportional to the value of loan, you can safely assume that the value of the loan is 1 unit and determine the ratio of the value for a particular service to the value of the loan. To obtain these ratios, change the value of the variable LOAN to 1 in the Payoff6 data set and invoke PROC DTREE again as follows:

```
   /* -- create the alternative PAYOFFS= data set  -- */
 data Payoff6a(drop=loan);
   length _STATE_ _ACT_ $24;
   loan=1;

   _ACT_='Deny loan';    _VALUE_=loan*0.08;   output;
   _STATE_='Pay off';    _VALUE_=loan*0.08;   output;
   _STATE_='Default';    _VALUE_=loan*0.08;   output;

   _ACT_='Approve loan';
   _STATE_='Pay off';    _VALUE_=loan*0.15;   output;
   _STATE_='Default';    _VALUE_=-1.0*loan;   output;
run;

   /* -- PROC DTREE statements                     -- */
title 'Loan Grant Decision';

proc dtree
     stagein=Stage6 probin=Prob6 payoffs=Payoff6a
     nowarning;

   evaluate / summary target=investigation;

   save;
   move payment before investigation;
   evaluate;

   recall;
   modify payment type;
   evaluate;

quit;
```

The optimal decision summary given in Output 7.5.5 shows that the ratio of the value of investigation that the loan officer currently engages in to the value of the loan is $0.1242 - 0.0909 = 0.0333$ to 1.

Output 7.5.5 Summary of the Loan Grant Decision with 1 Unit Loan

Loan Grant Decision

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Investigation	Decision
Recommendation	Chance
Application	Decision
Payment	Chance
ENDST	End

Output 7.5.5 *continued*

Decision Parameters	
Decision Criterion:	Maximize Expected Value (MAXEV)
Optimal Decision Yields:	0.1242

Optimal Decision Policy
Up to Stage Investigation

Alternatives or Outcomes	Cumulative Reward	Evaluating Value
Order investigation		0.1242*
Do not order		0.0909

The following messages are written to the SAS log:

```
NOTE: Present order of stages:

      Investigation(D), Recommendation(C), Application(D),
      Payment(C), _ENDST_(E).

NOTE: The current problem has been successfully saved.

NOTE: Present order of stages:

      Payment(C), Investigation(D), Recommendation(C),
      Application(D), _ENDST_(E).

NOTE: The currently optimal decision yields 0.1464.

NOTE: The original problem has been successfully recalled.

NOTE: Present order of stages:

      Investigation(D), Recommendation(C), Application(D),
      Payment(C), _ENDST_(E).

NOTE: The type of stage Payment has been changed.

NOTE: The currently optimal decision yields 0.15.
```

The preceding messages show that the ratio of the value of perfect investigation to the value of a loan is $0.1464 - 0.0909 = 0.0555$ to 1, and the ratio of the maximum amount the officer should pay for perfect control to the value of loan is $0.15 - 0.0909 = 0.591$ to 1.

Output 7.5.6, produced by the following statements, shows a table of the values of the investigation currently engaged in, the values of perfect investigation, and the values of perfect control for loans ranging from $10,000 to $100,000.

```
   /* create the data set for value of loan   */
   /* and corresponding values of services    */
data Datav6(drop=k ratio1 ratio2 ratio3);
```

```
   label loan="Value of Loan"
         vci="Value of Current Credit Investigation"
         vpi="Value of Perfect Credit Investigation"
         vpc="Value of Perfect Collecting Service";

      /* calculate ratios */
   ratio1=0.1242-0.0909;
   ratio2=0.1464-0.0909;
   ratio3=0.15-0.0909;

   Loan=0;
   do k=1 to 10;

         /* set the value of loan */
      loan=loan+10000;

         /* calculate the values of various services */
      vci=loan*ratio1;
      vpi=loan*ratio2;
      vpc=loan*ratio3;

         /* output current observation */
      output;
   end;
run;

   /* print the table of the value of loan    */
   /* and corresponding values of services    */
title 'Value of Services by Value of Loan';

proc print label;
   format loan vci vpi vpc dollar12.0;
run;
```

Output 7.5.6 Values of Loan and Associated Values of Service

Value of Services by Value of Loan

Obs	Value of Loan	Value of Current Credit Investigation	Value of Perfect Credit Investigation	Value of Perfect Collecting Service
1	$10,000	$333	$555	$591
2	$20,000	$666	$1,110	$1,182
3	$30,000	$999	$1,665	$1,773
4	$40,000	$1,332	$2,220	$2,364
5	$50,000	$1,665	$2,775	$2,955
6	$60,000	$1,998	$3,330	$3,546
7	$70,000	$2,331	$3,885	$4,137
8	$80,000	$2,664	$4,440	$4,728
9	$90,000	$2,997	$4,995	$5,319
10	$100,000	$3,330	$5,550	$5,910

Example 7.6: Petroleum Distributor's Decision Problem

The president of a petroleum distribution company currently faces a serious problem. His company supplies refined products to its customers under long-term contracts at guaranteed prices. Recently, the price for petroleum has risen substantially and his company will lose $450,000 this year because of its long-term contract with a particular customer. After a great deal of discussion with his legal advisers and his marketing staff, the president learns that the contract contains a clause that may be beneficial to his company. The clause states that when circumstances are beyond its control, the company may ask its customers to pay the prevailing market prices for up to 10% of the promised amount.

Several scenarios are possible if the clause is invoked. If the customer accepts the invocation of the clause and agrees to pay the higher price for the 10%, the company would turn a loss of $450,000 into a net profit of $600,000. If the customer does not accept the invocation, the customer may sue for damages or accept a settlement of $900,000 (resulting in a loss of $400,000) or simply decline to press the issue. In any case, the distribution company could then sell the 10% on the open market for an expected value of $500,000. However, the lawsuit would result in one of three possible outcomes: the company wins and pays no damages; the company loses and pays normal damages of $1,500,000; the company loses and pays double damages of $3,000,000. The lawyers also feel that this case might last three to five years if the customer decides to sue the company. The cost of the legal proceedings is estimated as $30,000 for the initial fee and $20,000 per year. The likelihood of the various outcomes are also assessed and reported as in Table 7.25. Suppose that the company decides to use a discount rate of 10% to determine the present value of future funds.

Table 7.25 Likelihood of the Outcomes in the Petroleum Distributor's Decision

Uncertainty	Outcome	Probability
Customer's Response	Accept the Invocation	0.1
	Reject the Invocation	0.9
Customer's Action if the Invocation is being Rejected	Press the Issue	0.1
	Settle the Case	0.45
	Sue for Damages	0.45
Case Last	3 Years	0.3
	4 Years	0.4
	5 Years	0.3
Lawsuit Result	Pay No Damages	0.15
	Pay Normal Damages	0.65
	Pay Double Damages	0.2

The structure for this decision problem is given in the STAGEIN= data set named Stage7.

```
/* -- create the STAGEIN= data set                              -- */
data Stage7;
   format _OUTCOM1 $14. _OUTCOM2 $14. ;
   input _STNAME_ $ _STTYPE_ $ _OUTCOM1 $
       _SUCC1 $ _OUTCOM2 $ _SUCC2 $ ;
   datalines;
Action    D   Invoking        Response  Not_Invoking    .
Response  C   Accept          .         Refuse          Lawsuit
Lawsuit   C   Press_Issue     .         Settle          .
```

```
.          .    Sue             Last      .                .
Last       C    3_Years         Result    4_Years          Result
.          .    5_Years         Result    .                .
Result     C    No_Damages      .         Normal_Damages   .
.          .    Double_Damages  .         .                .
;
```

The PROBIN= data set Prob7 contains the probability distributions for the chance nodes.

```
/* -- create the PROBIN= data set                      -- */
data Prob7;
   format _EVENT1_ _EVENT2_ $14.;
   input _EVENT1_ $ _PROB1_ _EVENT2_ $ _PROB2_ ;
   datalines;
Accept          0.1     Refuse          0.9
Press_Issue     0.1     Settle          0.45
Sue             0.45    .               .
3_Years         0.3     4_Years         0.4
5_Years         0.3     .               .
No_Damages      0.15    Normal_Damages  0.65
Double_Damages  0.20    .               .
;
```

The PAYOFFS= data set Payoff7 defines the payoffs for the various scenarios.

```
   /* -- create the PAYOFFS= data set            -- */
data Payoff7(drop=i j k D PCOST);
   length _ACTION_ _STATE1-_STATE4 $16;

      /* possible outcomes for the case last        */
   array YEARS{3}   $16. _TEMPORARY_ ('3_Years',
                                      '4_Years',
                                      '5_Years' );

      /* numerical values for the case last  */
   array Y{3}            _TEMPORARY_ (3, 4, 5);

      /* possible outcomes for the size of judgment */
   array DAMAGES{3} $16. _TEMPORARY_ ('No_Damages',
                                      'Normal_Damages',
                                      'Double_Damages' );

      /* numerical values for the size of judgment  */
   array C{3}            _TEMPORARY_ (0, 1500, 3000);

   D=0.1;                          /* discount rate */

      /* payoff for the scenario which the          */
      /* 10 percent clause is not invoked           */
   _ACTION_='Not_Invoking';   _VALUE_=-450;   output;

      /* the clause is invoked */
   _ACTION_='Invoking';
```

```
      /* payoffs for scenarios which the clause is  */
      /* invoked and the customer accepts the       */
      /* invocation                                 */
   _STATE1='Accept';           _VALUE_=600;   output;

      /* the customer refuses the invocation        */
   _STATE1='Refuse';

      /* payoffs for scenarios which the clause is  */
      /* invoked and the customer refuses the       */
      /* invocation but decline to press the issue  */
   _STATE2='Press_Issue';   _VALUE_=500;      output;

      /* payoffs for scenarios which the clause is  */
      /* invoked and the customer refuses the       */
      /* invocation but willing to settle out of    */
      /* court for 900K                             */
   _STATE2='Settle';        _VALUE_=500-900;   output;

      /* the customer will sue for damages          */
   _STATE2='Sue';
   do i=1 to 3;
      _STATE3=YEARS{i};

         /* determine the cost of proceedings       */
      PCOST=30;  /* initial cost of the proceedings */

         /* additional cost for every years in      */
         /* in present value                        */
      do k=1 to Y{i};
         PCOST=PCOST+(20/((1+D)**k));
      end;

         /* loop for all poss. of the lawsuit result */
      do j=1 to 3;
         _STATE4=DAMAGES{j}; /* the damage have to paid */

            /* compute the net return in present value  */
         _VALUE_=500-PCOST-(C{j}/((1+D)**Y{i}));

            /* output an observation for the payoffs */
            /* of this scenario                      */
         output;
      end;
   end;

run;

   /* -- print the payoff table                      -- */
title "Petroleum Distributor's Decision";
title3 "Payoff Table";

proc print;
run;
```

The payoff table of this problem is displayed in Output 7.6.1.

Output 7.6.1 Payoffs for the Petroleum Distributor's Problem

Petroleum Distributor's Decision

Payoff Table

Obs	_ACTION_	_STATE1	_STATE2	_STATE3	_STATE4	_VALUE_
1	Not_Invoking					-450.00
2	Invoking	Accept				600.00
3	Invoking	Refuse	Press_Issue			500.00
4	Invoking	Refuse	Settle			-400.00
5	Invoking	Refuse	Sue	3_Years	No_Damages	420.26
6	Invoking	Refuse	Sue	3_Years	Normal_Damages	-706.71
7	Invoking	Refuse	Sue	3_Years	Double_Damages	-1833.68
8	Invoking	Refuse	Sue	4_Years	No_Damages	406.60
9	Invoking	Refuse	Sue	4_Years	Normal_Damages	-617.92
10	Invoking	Refuse	Sue	4_Years	Double_Damages	-1642.44
11	Invoking	Refuse	Sue	5_Years	No_Damages	394.18
12	Invoking	Refuse	Sue	5_Years	Normal_Damages	-537.20
13	Invoking	Refuse	Sue	5_Years	Double_Damages	-1468.58

Note that the payoffs of the various scenarios in Output 7.6.1 are in thousands of dollars and are *net present values* (NPV) (Baird 1989). For example, the payoff for the following scenario "invoking the clause; the customer refuses to accept this and sues for damages; the case lasts four years and the petroleum distribution company loses and pays double damages" is calculated as

$$\begin{aligned} \text{Payoff} &= 500 - \text{NPV of proceedings cost} \\ &\quad\; - \text{NPV of damages of } 3,000,000 \\ &= -1642.44 \end{aligned}$$

where

$$\text{NPV of proceedings cost} = 30 + \sum_{k=1}^{4} 20/(1+0.1)^k$$

and

$$\text{NPV of damages of } 3,000,000 = 3000/(1+0.1)^4$$

This is because the company can sell the 10% for $500,000 immediately and pay the $3,000,000 damages four years from now. The net present value of the proceedings is determined by paying the $30,000 initial fee now and a fee of $20,000 after every year up to four years. The value of 0.1 is the discount rate used.

The following statements evaluate the problem and plot the optimal solution.

```
   /* -- define graphics options                          -- */
goptions colors=(green red blue);
goptions hsize=8 in vsize=8.4 in;

   /* -- define title                                    -- */
title h=2.5 "Petroleum Distributor's Decision";

   /* -- PROC DTREE statements                           -- */
proc dtree stagein=Stage7 probin=Prob7 payoffs=Payoff7;
   evaluate / summary;
   treeplot / graphics compress nolg name="dt6p1" ftext='Cumberland AMT'
           ybetween=1 cell lwidth=2 lwidthb=3 hsymbol=3;
quit;
```

The optimal decision summary in Output 7.6.2 suggests that the president should invoke the 10% clause because it would turn a loss of $450,000 into an expected loss of $329,000 in present value.

Output 7.6.2 Summary of the Petroleum Distributor's Decision

Petroleum Distributor's Decision

The DTREE Procedure

Optimal Decision Summary

Order of Stages	
Stage	**Type**
Action	Decision
Response	Chance
Lawsuit	Chance
Last	Chance
Result	Chance
ENDST	End

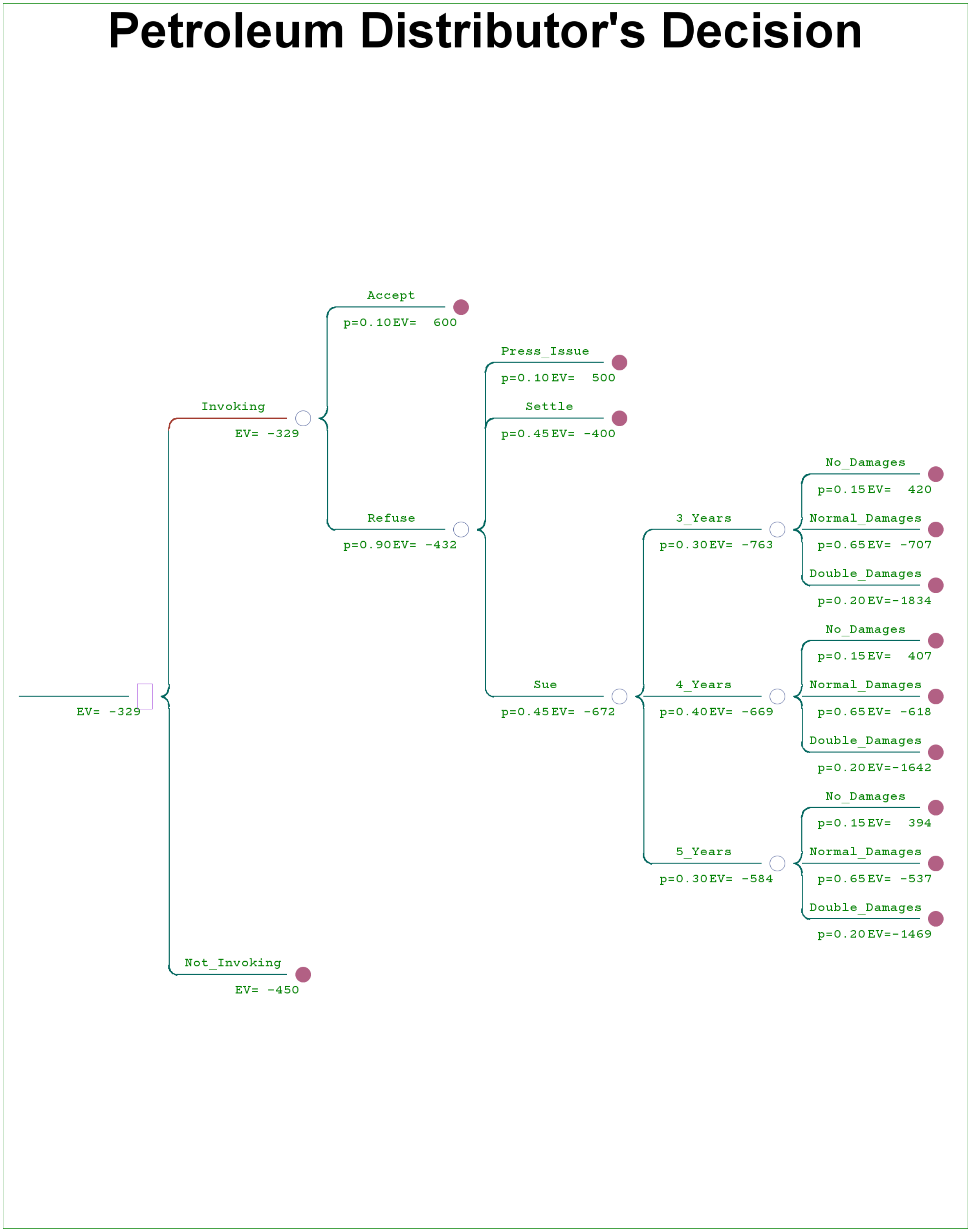
Decision Parameters	
Decision Criterion:	Maximize Expected Value (MAXEV)
Optimal Decision Yields:	-329

Optimal Decision Policy
Up to Stage Action

Alternatives or Outcomes	Cumulative Reward	Evaluating Value
Invoking		-329*
Not_Invoking		-450

The decision tree for this problem is shown in Output 7.6.3. There you can find the expected value of each scenario.

Output 7.6.3 Decision Tree for the Petroleum Distributor's Decision



The president feels that the estimated likelihood of lawsuit outcomes is fairly reliable. However, the assessment of the likelihood of the customer's response and reaction is extremely difficult to estimate. Because of this, the president would like to keep the analysis as general as possible. His staff suggests using the symbols p and q to represent the probability that the customer will accept the invocation and the probability that the customer will decline to press the issue if he refuses the invocation, respectively. The probabilities of the other possible outcomes about the customer's response and reaction to the invocation of the 10% clause are listed in Table 7.26.

Table 7.26 Probabilities of the Petroleum Distributor's Decision

Uncertainty	Outcome	Probability
Customer's Response	Accept the Invocation	p
	Reject the Invocation	1 - p
Customer's Action if the Invocation is being Rejected	Press the Issue	q
	Settle the Case	$(1-q)/2$
	Sue for Damages	$(1-q)/2$

Now from the decision tree shown in Output 7.6.3, the expected value of the outcome `'Refuse'` is

$$\begin{aligned}\text{EV} &= 500q - 400(1-q)/2 - 672(1-q)/2 \\ &= 500q - 200 + 200q - 336 + 336q \\ &= 1036q - 536\end{aligned}$$

Hence, the expected payoff if the petroleum distribution company invokes the clause is

$$\begin{aligned}\text{EV} &= 600p + (1036q - 536)(1-p) \\ &= 1136p + 1036q - 1036pq - 536 \\ &= 1136p + 1036(1-p)q - 536\end{aligned}$$

Therefore, the president should invoke the 10% clause if

$$1136p + 1036(1-p)q - 536 > -450$$

or

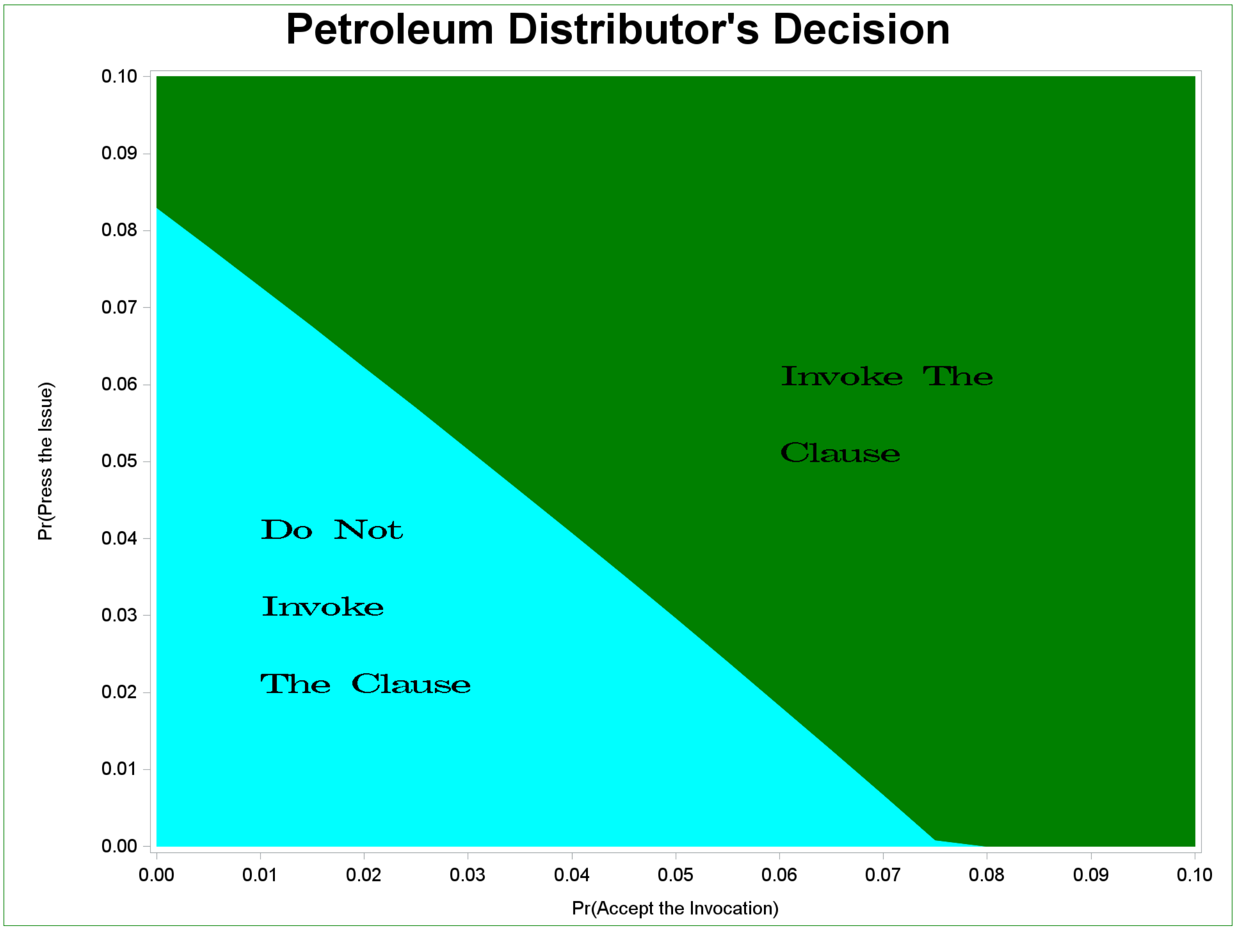
$$q > \frac{86 - 1136p}{1036 - 1036p}$$

This result is depicted in Output 7.6.4, which is produced by the following statements:

```
   /* -- create data set for decision diagram        -- */
data Data7(drop=i);
   P=0.0;                  /* initialize P */

      /* loop for all possible values of P */
   do i=1 to 21;

         /* determine the corresponding Q  */
      Q=(86-(1136*P))/(1036*(1.0-P));
      if Q < 0.0 then Q=0.0;

         /* output this data point */
      output;

         /* set next possible value of P   */
      P=P+0.005;
   end;

run;

   /* create the ANNOTATE= data set for labels of  */
   /* decision diagram                             */
data label;
   length FUNCTION STYLE COLOR $8;
   length XSYS YSYS            $1;
   length WHEN POSITION        $1;
   length X Y                   8;
   length SIZE ROTATE           8;

   WHEN = 'A';
   POSITION='0';
   XSYS='2';
   YSYS='2';
   input FUNCTION $ X Y STYLE $ SIZE COLOR $
         ROTATE TEXT $ & 16.;
   datalines;
label   0.01    0.04    centx   2       black   .    Do Not
label   0.01    0.03    centx   2       black   .    Invoke
label   0.01    0.02    centx   2       black   .    The Clause
label   0.06    0.06    centx   2       black   .    Invoke The
label   0.06    0.05    centx   2       black   .    Clause
;

   /* -- define symbol characteristics for boundary  -- */
symbol1 i=joint v=NONE l=1 ci=black;

   /* -- define pattern for area fill                -- */
pattern1 value=msolid color=cyan;
pattern2 value=msolid color=green;

   /* -- define axis characteristics                 -- */
axis1 label=('Pr(Accept the Invocation)')
```

```
      order=(0 to 0.1 by 0.01) minor=none;
axis2 label=(angle=90 'Pr(Press the Issue)')
      order=(0 to 0.1 by 0.01) minor=none;

   /* -- plot decision diagram                      -- */
title h=2.5 "Petroleum Distributor's Decision";
proc gplot data=Data7 ;
   plot Q*P=1 / haxis=axis1
                vaxis=axis2
                annotate=label
                name="dt6p2"
                frame
                areas=2;
run;
quit;
```

Output 7.6.4 Decision Diagram for the Petroleum Distributor's Problem



The decision diagram in Output 7.6.4 is an analysis of the sensitivity of the solution to the probabilities that the customer will accept the invocation and that the customer will decline to press the issue. He should invoke the clause if he feels the customer's probabilities of outcomes **`'Accept'`** and **`'Press_Issue'`**, p and q, are located in the upper-right area marked as **`'Invoke The Clause'`** in Output 7.6.4 and should not invoke the clause otherwise. Note that the values $p = 0.1$ and $q = 0.1$ used in this example are located on the upper right corner on the diagram.

Statement and Option Cross-Reference Tables

The following tables reference the statements and options in the DTREE procedure (except the PROC DTREE statement and the QUIT statement) that are illustrated by the examples in this section.

Table 7.27 Statements Specified in Examples

Statement	1	2	3	4	5	6
EVALUATE	X	X	X	X	X	X
MODIFY					X	
MOVE					X	
RECALL					X	
RESET	X					
SAVE					X	
SUMMARY	X	X			X	
TREEPLOT			X	X		X
VARIABLES	X					
VPC					X	
VPI					X	

Table 7.28 Options Specified in Examples

Option	1	2	3	4	5	6
ANNOTATE=				X		
COMPRESS			X	X		X
CRITERION=	X	X		X		
EVENT=	X					
FTEXT=			X			X
GRAPHICS			X	X		X
HSYMBOL=			X			X
LINKA=				X		
LINKB=				X		
LINKC=						
LSTYLEB=			X			
LWIDTH=			X			X
LWIDTHB=			X			X
NAME=			X	X		X
NOLEGEND				X		X
NOWARNING	X	X	X		X	
OUTCOME=	X					
PAYOFFS=	X	X	X	X	X	X
PROB=	X					
PROBIN=	X	X	X	X	X	X
REWARD=	X					
RT=	X	X		X		
STAGE=	X					
STAGEIN=	X	X	X	X	X	X
STATE=	X					
SUCCESSOR=	X					
SUMMARY					X	X
SYMBOLC=				X		
SYMBOLD=				X		
SYMBOLE=				X		
TARGET=	X	X			X	
TYPE=	X					
VALUE=						
YBETWEEN=			X			X

References

Baird, B. F. (1989). *Managerial Decisions under Uncertainty: An Introduction to the Analysis of Decision Making*. New York: John Wiley & Sons.

Howard, R. A. (1968). "The Foundations of Decision Analysis." *IEEE Transactions on System Science and Cybernetics* SSC-4:211–219.

Howard, R. A. (1988). "Decision Analysis: Practice and Promise." *Management Science* 34:679–695.

Kuhfeld, W. F. (2010). *Statistical Graphics in SAS: An Introduction to the Graph Template Language and the Statistical Graphics Procedures*. Cary, NC: SAS Institute Inc.

Raiffa, H. (1970). *Decision Analysis Introductory Lectures on Choices under Uncertainty*. Reading, MA: Addison-Wesley.

Chapter 8

The GANTT Procedure

Contents

Overview: GANTT Procedure

The GANTT procedure produces a Gantt chart, which is a graphical scheduling tool for the planning and control of a project. In its most basic form, a Gantt chart is a bar chart that plots the tasks of a project versus time. PROC GANTT displays a Gantt chart that corresponds to a project schedule such as that produced by the CPM procedure or one that is input directly to the procedure. PROC GANTT offers several options and statements for tailoring the chart to your needs.

Using PROC GANTT, you can plot the predicted early and late schedules and identify critical, supercritical, and slack activities. In addition, you can visually monitor a project in progress with the actual schedule and compare the actual schedule against a target baseline schedule. You can also graphically view the effects of scheduling a project subject to resource limitations. Any combination of these schedules can be viewed simultaneously (provided the relevant data exist) together with any user-specified variables of interest, such as project deadlines and other important dates. PROC GANTT enables you to display the early, late, and actual schedules in a single bar to produce a more meaningful schedule for tracking an activity in progress.

PROC GANTT can display the project logic on the Gantt chart by exhibiting dependencies between tasks by using directed arcs to link the related activities. You can use either the activity-on-arc (AOA) or Activity-on-Node (AON) style of input for defining the project network. In addition, the GANTT procedure recognizes nonstandard precedence types. With PROC GANTT, you can display weekends, holidays, and multiple calendars, and you can depict milestones, reference lines, and a timenow line on the chart. PROC GANTT enables you to annotate text and graphics on the Gantt chart and provides you with a wide variety of options to control and customize the graphical appearance of the chart.

The GANTT procedure also supports an automatic text annotation facility that is designed specifically for labeling Gantt charts independently of the SAS/GRAPH Annotate facility. It enables you to display label strings with a minimum of effort and data entry while providing the capability for more complex chart labeling situations. An important feature of this facility is the ability to link label coordinates and text strings to variables in the Schedule data set. This means that you can preserve the Label data set even though the schedule dates may change. Several options enable you to customize the annotation, such as the clipping of text strings that run off the page or the chart and the specification of a split character to split labels that are too long.

Using the GANTT procedure, you can produce a wide variety of Gantt charts. You can generate zoned Gantt charts with several options to control its appearance. You can display a zone variable column as well as draw a line demarcating the different zones. You can also control the bar height and bar offset of each type of schedule bar. This enables you to change the display order of the schedules as well as giving you the capability to produce a Gantt chart with embedded bars. You can override the default schedule bar pattern assignments at the activity level. In addition, you can restrict the schedule types to which the specified pattern is to be applied. You can also override the text color for selected columns of activity text at the activity level. These features facilitate the production of multiproject and multiprocess Gantt charts. Finally, you can also associate HTML pages with activity bars and create Web-enabled Gantt charts.

The GANTT procedure enables you to control the number of pages output by the procedure in both horizontal and vertical directions. In addition, you can control the number of jobs displayed per page as well as the number of tickmarks displayed per page. You can display ID variables on every page and even let the procedure display the maximum number of ID variables that can fit on one page. You can number the pages, justify the Gantt chart in the horizontal and vertical directions with respect to the page boundaries, and maintain the original aspect ratio of the Gantt chart on each page.

PROC GANTT gives you the option of displaying the Gantt chart in one of three modes: line-printer, full-screen, or graphics mode. The default mode is graphics mode, which enables you to produce charts of high resolution quality. Graphics mode requires SAS/GRAPH software. See the section “Graphics Version” on page 536 for more information on producing high-quality Gantt charts. You can also produce line-printer quality Gantt charts by specifying the LINEPRINTER option in the PROC GANTT statement. In addition to submitting the output to either a plotter or printer, you can view the Gantt chart at the terminal in full-screen mode by specifying the FULLSCREEN option in the PROC GANTT statement. See the section “Full-Screen Version” on page 532 for more information on viewing Gantt charts in full-screen mode. The GANTT procedure also produces a macro variable that indicates the status of the invocation and also contains other useful statistics about the Gantt charts generated by the invocation.

There are several distinctive features that characterize the appearance of the chart produced by the GANTT procedure:

- The horizontal axis represents time, and the vertical axis represents the sequence of observations in the data set.
- Both the time axis and the activity axis can be plotted across more than one page.
- The procedure automatically provides extensive labeling of the time axis, enabling you to determine easily the exact time of events plotted on the chart. The labels are determined on the basis of the formats of the times being plotted. You can also specify user-defined formats for the labeling.

- In graphics mode, the COMPRESS option in the CHART statement enables you to produce the entire Gantt chart on one page. The PCOMPRESS option enables you to produce the entire Gantt chart on one page while maintaining the original aspect ratio of the Gantt chart. Both these options work in conjunction with the HPAGES= and VPAGES= options, which specify the number of pages in the horizontal and vertical directions for the chart.

Project information is communicated into PROC GANTT using SAS data sets. The input data sets used by PROC GANTT are as follows:

- **The Schedule data set** contains the early, late, actual, resource-constrained, and baseline schedules and any other activity-related information. The activity-related information can include precedence information, calendar used by the activity, special dates, and any other information that you want to identify with each activity. This data set can be the same as the Schedule data set produced by PROC CPM, or it can be created separately by a DATA step. Each observation in the Schedule data set represents an activity and is plotted on a separate row of the chart unless activity splitting during resource-constrained scheduling has caused an activity to split into disjoint segments. For details regarding the output format in this case, see the section "Displayed Output" on page 559.

- **The Precedence (Logic) data set** contains the precedence information of the project in AON format in order to draw a Logic Gantt chart of the project. Specifying this data set is not necessary if the precedence information exists in the Schedule data set. If the data set is specified, however, the ACTIVITY variable must exist in both the Schedule and Precedence data sets.

  Typically you would use this feature when scheduling in PROC CPM with nonstandard precedence constraints where the LAG variables are not transferred to the Schedule data set or with the COLLAPSE option. Setting the Precedence data set for PROC GANTT to be the Activity data set (used in PROC CPM) establishes the required precedence relationships. This is also a convenient feature when drawing several Gantt charts for the same project with different schedule information (such as when monitoring a project in progress). Specifying a Precedence data set avoids having to duplicate the precedence information in every Schedule data set.

- **The Label data set** contains the label information of the project that enables you to draw labeled Gantt charts independently of the SAS/GRAPH Annotate facility. It requires a minimum of effort and provides you with a convenient mechanism to link label strings and their coordinates to variables in the Schedule data set. Another convenient feature is its ability to replicate labels across all activities. Both these features facilitate reuse of the Label data set.

- **The Workday and the Calendar data sets** together enable you to represent any type of work pattern, during a week and within each day of the week, on the Gantt chart. The same Workday and Calendar data sets used by PROC CPM can also be passed to PROC GANTT.

- **The Holiday data set** enables you to associate standard holidays and vacation periods with each calendar and represent them on the Gantt chart. Like the Workday and Calendar data sets, the same Holiday data set used by PROC CPM can also be used by PROC GANTT.

- **The Annotate data set** contains the graphics and text that are to be annotated on the Gantt chart. This data set is used by the GANTT procedure in conjunction with the Annotate facility in SAS/GRAPH software.

The GANTT procedure produces one output data set.

- **The Imagemap data set** contains the outline coordinates for the schedule bars used in the Gantt chart that can be used to generate HTML MAP tags.

When displaying the precedence relationships between activities on the Gantt chart, bear in mind the following facts with regard to data sets used by PROC GANTT:

- The Schedule data set (and optionally the Precedence data set) contains the variables that define the precedence relationships between activities in the project.
- You can handle nonstandard precedence constraints in PROC GANTT when using AON format by identifying the LAG variables in the CHART statement.
- When you use PROC CPM to produce the schedule for a project with nonstandard precedence relationships, the LAG variables are not automatically included in the Schedule data set. Use an ID statement or the XFERVARS option in the PROC CPM statement to add them.
- When you generate the schedule using PROC CPM with the COLLAPSE option, it is recommended that you use the Activity data set to define the precedence relationships for the Gantt procedure by specifying the PRECDATA= option in the PROC GANTT statement. This ensures that all the relevant precedence information is extracted.

Each option and statement available in the GANTT procedure is explained in the section "Syntax: GANTT Procedure" on page 494. The section "Examples: GANTT Procedure" on page 567 illustrates most of these options and statements.

Getting Started: GANTT Procedure

In order to draw a Gantt chart, at the very minimum you need a Schedule data set. This data set is expected to be *similar* to the OUT= Schedule data set produced by PROC CPM, with each observation representing an activity in the project. It is possible to obtain a detailed Gantt chart by specifying the following single statement:

PROC GANTT *DATA= SAS-data-set* **;**

The data set specified is the Schedule data set produced by PROC CPM.

As an example of this, consider the software development project in the "Getting Started" section in Chapter 4, "The CPM Procedure." The output schedule for this example is saved in a data set, INTRO1, which is displayed in Figure 8.1.

Figure 8.1 Software Project Plan

Project Schedule

Obs	activity	succesr1	succesr2	duration	descrpt
1	TESTING	RECODE		20	Initial Testing
2	PRELDOC	DOCEDREV	QATEST	15	Prel. Documentation
3	MEETMKT	RECODE		1	Meet Marketing
4	RECODE	DOCEDREV	QATEST	5	Recoding
5	QATEST	PROD		10	QA Test Approve
6	DOCEDREV	PROD		10	Doc. Edit and Revise
7	PROD			1	Production

Obs	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	01MAR04	20MAR04	01MAR04	20MAR04	0	0
2	01MAR04	15MAR04	11MAR04	25MAR04	10	10
3	01MAR04	01MAR04	20MAR04	20MAR04	19	19
4	21MAR04	25MAR04	21MAR04	25MAR04	0	0
5	26MAR04	04APR04	26MAR04	04APR04	0	0
6	26MAR04	04APR04	26MAR04	04APR04	0	0
7	05APR04	05APR04	05APR04	05APR04	0	0

The following code produces the Gantt chart shown in Figure 8.2.

```
title 'Line-Printer Gantt Chart';
proc gantt lineprinter data=intro1;
run;
```

The DATA= option could be omitted if the INTRO1 data set is the most recent data set created; by default, PROC GANTT uses the _LAST_ data set.

Figure 8.2 Line-Printer Gantt Chart

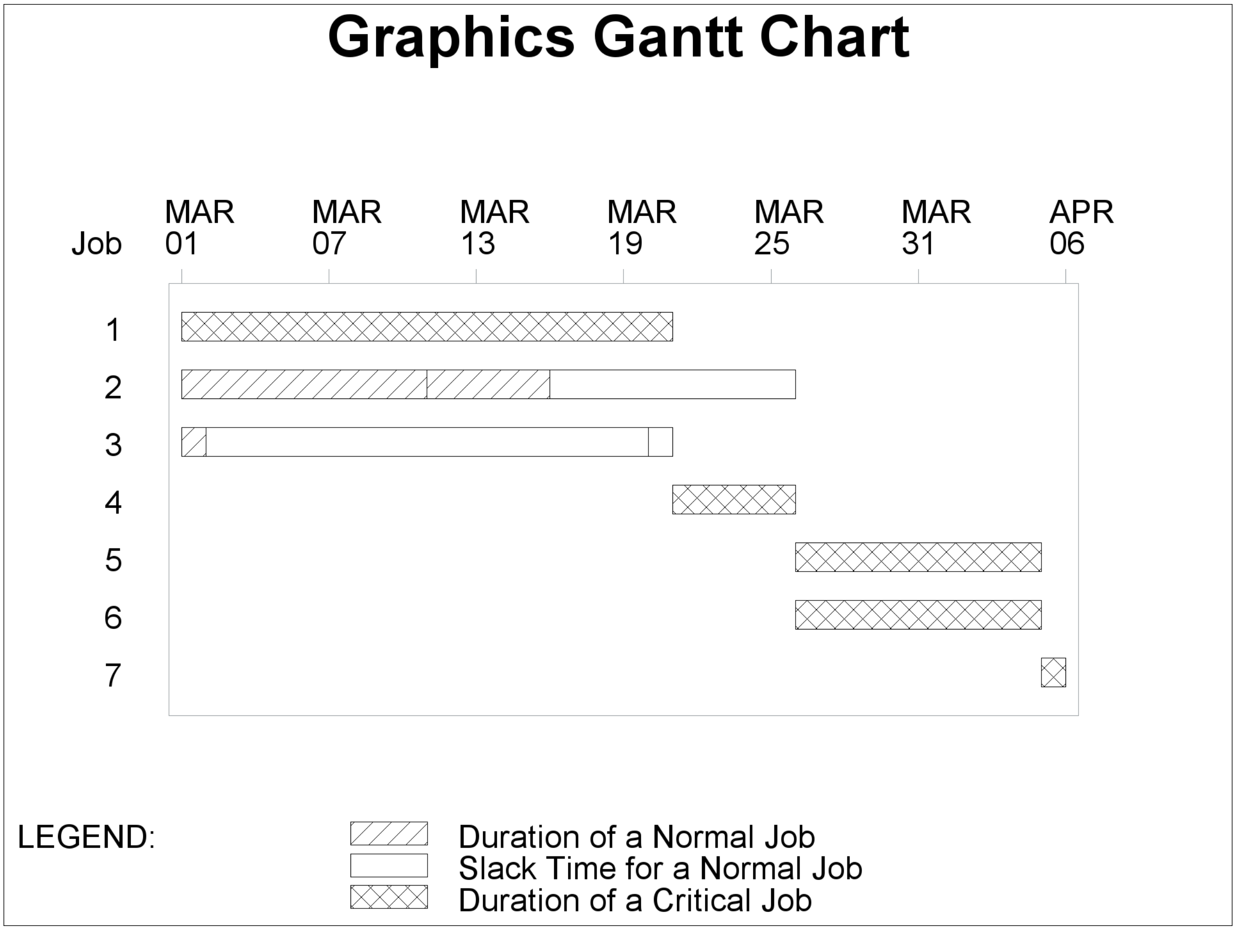
Line-Printer Gantt Chart

```
       MAR   MAR   MAR   MAR   MAR   MAR   APR
Job    01 04 07 10 13 16 19 22 25 28 31 03 06
      -+--+--+--+--+--+--+--+--+--+--+--+--+--+-
      |                                      |
  1   |*===================*                 |
      |                                      |
  2   |<---------<---->.........>            |
      |                                      |
  3   |<>.................<>                 |
      |                                      |
  4   |                     *====*           |
      |                                      |
  5   |                         *=========*  |
      |                                      |
  6   |                         *=========*  |
      |                                      |
  7   |                                    **|
      |                                      |
      -+--+--+--+--+--+--+--+--+--+--+--+--+--+-


                    LEGEND


    Symbol     Explanation

    <---->     Duration of a Normal Job

    >....>     Slack Time for a Normal Job

    *====*     Duration of a Critical Job
```

You can produce a high-resolution graphics quality Gantt chart by specifying the GRAPHICS option instead of the LINEPRINTER option in the PROC GANTT statement. Graphics mode is also the default display mode. The resulting Gantt chart is shown in Figure 8.3.

```
proc gantt graphics data=intro1;
run;
```

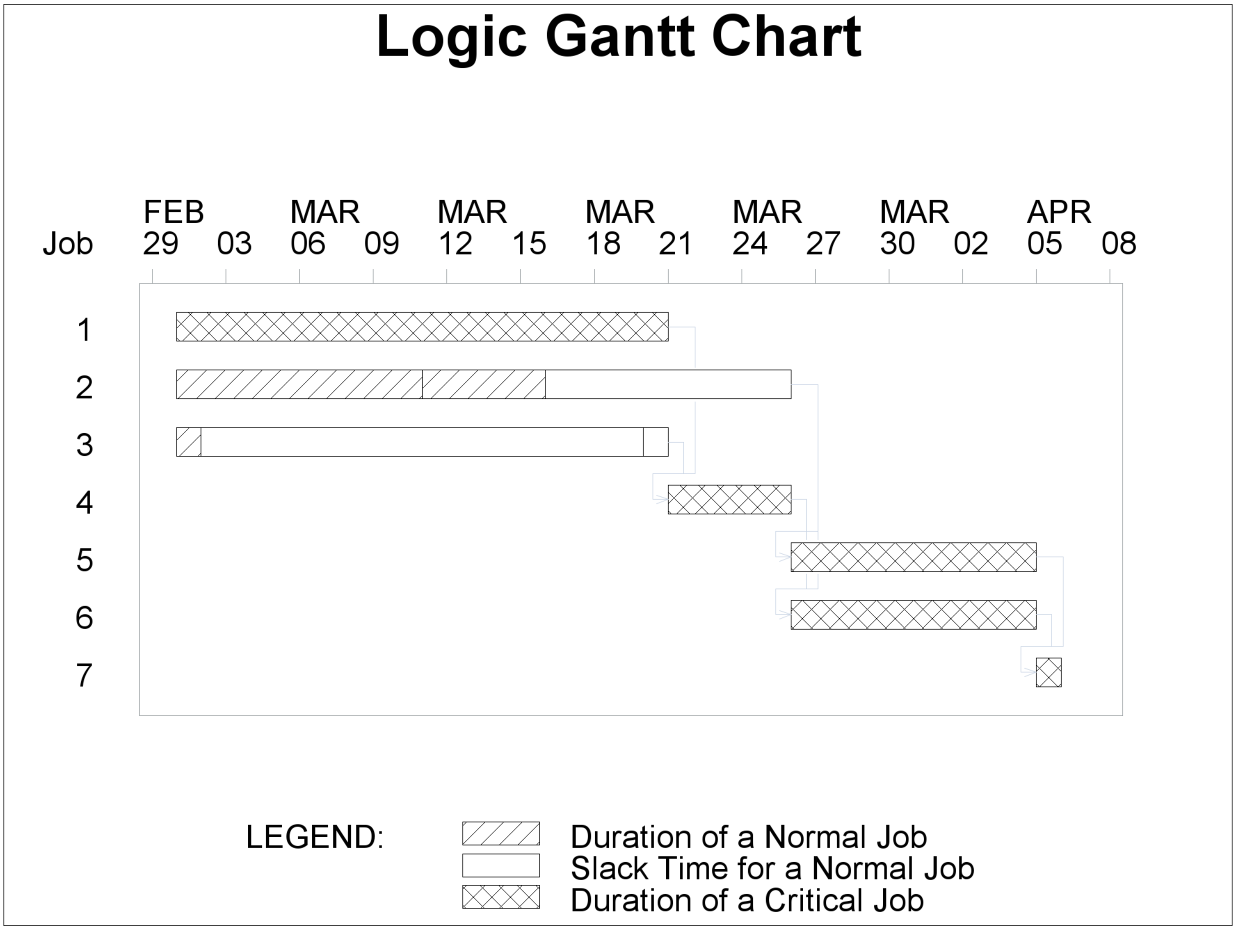
Figure 8.3 Graphics Gantt Chart



Finally, you can draw a Logic Gantt chart by defining the precedence information to PROC GANTT in AON format using the ACTIVITY= and SUCCESSOR= options in the CHART statement. The Logic Gantt chart is shown in Figure 8.4.

```
proc gantt data=intro1;
   chart / activity=activity successor=(succesr1-succesr2);
run;
```

Figure 8.4 Logic Gantt Chart



For further examples illustrating typical invocations of the GANTT procedure when managing projects, see Chapter 3, "Introduction to Project Management."

Syntax: GANTT Procedure

The following statements are used in PROC GANTT:

PROC GANTT *options* ;
BY *variables* ;
CHART *specifications / options* ;
ID *variables* ;

Functional Summary

Table 8.1 outlines the options available for the GANTT procedure, classified by function.

Table 8.1 Functional Summary

Description	Statement	Option
Axis Formatting Options		
Specifies the increment for labeling axis	CHART	INCREMENT=
Specifies the ending time for axis	CHART	MAXDATE=
Specifies the starting time for axis	CHART	MINDATE=
Specifies the smallest interval identified on chart	CHART	MININTERVAL=
Suppresses time portion of datetime tickmark	CHART	NOTMTIME
Specifies the number of columns per mininterval	CHART	SCALE=
Specifies the format of time axis labels	CHART	TIMEAXISFORMAT=
Uses first plot variable format for tickmarks	CHART	USEFORMAT
Bar Enhancement Options		
Specifies the actual bar height	CHART	ABARHT=
Specifies the actual bar offset	CHART	ABAROFF=
Specifies the default bar height	CHART	BARHT=
Specifies the default bar offset	CHART	BAROFF=
Specifies the baseline bar height	CHART	BBARHT=
Specifies the baseline bar offset	CHART	BBAROFF=
Specifies the color of connect line	CHART	CHCON=
Specifies the early/late bar height	CHART	EBARHT=
Specifies the early/late bar offset	CHART	EBAROFF=
Specifies the holiday bar height	CHART	HBARHT=
Specifies the holiday bar offset	CHART	HBAROFF=
Specifies the character for drawing connect line	CHART	HCONCHAR=
Draws a horizontal connect line	CHART	HCONNECT
Specifies the characters for drawing schedule	CHART	JOINCHAR=
Specifies the line style of connect line	CHART	LHCON=
Suppresses PATTERN variable for bar fills	CHART	NOPATBAR
Specifies the overprint character for schedule variables	CHART	OVERLAPCH=
Specifies the overprint character for CHART variables	CHART	OVPCHAR=
Specifies the schedule types that use the PATTERN variable	CHART	PATLEVEL=

Table 8.1 *continued*

Description	Statement	Option
Specifies the PATTERN variable for bar fills and text color	CHART	PATTERN=
Specifies the resource bar height	CHART	RBARHT=
Specifies the resource bar offset	CHART	RBAROFF=
Specifies the characters for plotting times	CHART	SYMCHAR=
Calendar Options		
Specifies the calendar identifier	CHART	CALID=
Specifies the length of the workday	CHART	DAYLENGTH=
Specifies the beginning of the workday	CHART	DAYSTART=
Marks all breaks in a day	CHART	MARKBREAK
Marks all non-working days	CHART	MARKWKND
Data Set Options		
Specifies the Annotate data set	GANTT	ANNOTATE=
	CHART	ANNOTATE=
Specifies the Calendar data set	GANTT	CALEDATA=
Specifies the Schedule data set	GANTT	DATA=
Specifies the Holiday data set	GANTT	HOLIDATA=
Specifies the Imagemap output data set	GANTT	IMAGEMAP=
Specifies the Label data set	GANTT	LABDATA=
Specifies the Precedence (Logic) data set	GANTT	PRECDATA=
Specifies the Work pattern data set	GANTT	WORKDATA=
Graphics Catalog Options		
Specifies the description of the catalog entry	CHART	DESCRIPTION=
Specifies the name of graphics catalog	GANTT	GOUT=
Specifies the name of catalog entry	CHART	NAME=
Holiday Options		
Specifies the character for plotting holidays	CHART	HOLICHAR=
Specifies the holiday start variable	CHART	HOLIDAY=
Specifies the holiday duration variable	CHART	HOLIDUR=
Specifies the holiday finish variable	CHART	HOLIFIN=
Specifies the holiday duration units	CHART	INTERVAL=
ID Variable Options		
Specifies the number of columns between ID variables	CHART	BETWEEN=
Marks critical activities	CHART	CRITFLAG
Specifies the activity text columns that use pattern color	CHART	CTEXTCOLS=
Allows duplicate ID values	CHART	DUPOK
Displays ID variables on every page	CHART	IDPAGES
Maximizes number of ID variables on page	CHART	MAXIDS
Suppresses job number	CHART	NOJOBNUM

Table 8.1 *continued*

Description	Statement	Option
Specifies the split character for dividing ID labels	GANTT	SPLIT=
Strips leading blanks from character variables	GANTT	STRIPIDBLANKS
Labeling Options		
Specifies the label variable linking to Schedule data set	CHART	LABVAR=
Specifies the rules for label layout	CHART	LABRULE=
Specifies the split character for labels	CHART	LABSPLIT=
Specifies the maximum number of digits in integer label	GANTT	LABMAXINT=
Logic Options		
Specifies the activity variable for AON format	CHART	ACTIVITY=
Uses AOA precedence specifications	CHART	AOA
Specifies the color of precedence connections	CHART	CPREC=
Specifies the headnode variable for AOA format	CHART	HEAD=
Specifies the lag variable for AON format	CHART	LAG=
Specifies the schedule bar connected to precedence lines	CHART	LEVEL=
Specifies the line style of precedence connections	CHART	LPREC=
Specifies the maximum displacement of local vertical	CHART	MAXDISLV=
Specifies the minimum interdistance of global verticals	CHART	MININTGV=
Specifies the minimum offset of global vertical	CHART	MINOFFGV=
Specifies the minimum offset of local vertical	CHART	MINOFFLV=
Suppresses drawing arrow head	CHART	NOARROWHEAD
Suppresses automatic range extension	CHART	NOEXTRANGE
Terminates procedure if bad precedence data	CHART	SHOWPREC
Specifies the successor variable for AON format	CHART	SUCCESSOR=
Specifies the tailnode variable for AOA format	CHART	TAIL=
Specifies the width of precedence connections	CHART	WPREC=
Milestone Options		
Specifies the milestone color	CHART	CMILE=
Specifies the duration variable	CHART	DUR=
Specifies the font for the milestone symbol	CHART	FMILE=
Specifies the milestone height	CHART	HMILE=
Specifies the milestone character	CHART	MILECHAR=
Specifies the value for the milestone symbol	CHART	VMILE=
Miscellaneous Options		
Invokes full-screen version	GANTT	FS
Invokes graphics version	GANTT	GRAPHICS
Invokes line-printer version	GANTT	LP
Specifies the maximum number of decimals for a number	GANTT	MAXDEC=
Specifies the unit for padding finish times	CHART	PADDING=
Specifies the upper limit on number of pages	CHART	PAGES=
Displays summary of symbols and patterns	CHART	SUMMARY

Table 8.1 *continued*

Description	Statement	Option
Page Layout Options		
Positions chart at bottom of page	CHART	BOTTOM
Specifies the axis color	CHART	CAXIS=
Specifies the frame fill color	CHART	CFRAME=
Specifies the width of the chart axis area	CHART	CHARTWIDTH=
Draws chart on one page in graphics mode	CHART	COMPRESS
Fills each page as much as possible	CHART	FILL
Specifies the characters for table outlines and dividers	CHART	FORMCHAR=
Specifies the number of pages spanning time axis	CHART	HPAGES=
Left justifies chart	CHART	LEFT
Specifies the line width	CHART	LWIDTH=
Specifies the number of activities on each page	CHART	NJOBS=
Suppresses frame	CHART	NOFRAME
Suppresses legend	CHART	NOLEGEND
Suppresses page number at upper right corner	CHART	NOPAGENUM
Specifies the number of tickmarks on each page	CHART	NTICKS=
Displays page number at upper right corner	CHART	PAGENUM
Draws chart proportionally on one page	CHART	PCOMPRESS
Right justifies chart	CHART	RIGHT
Specifies the number of rows between consecutive activities	CHART	SKIP=
Positions chart at top of page	CHART	TOP
Specifies the number of pages spanning activity axis	CHART	VPAGES=
Reference Line Options		
Specifies the reference line color	CHART	CREF=
Specifies the reference line style	CHART	LREF=
Specifies the placement of the reference lines	CHART	REF=
Specifies the reference line character	CHART	REFCHAR=
Specifies that reference lines should be labeled	CHART	REFLABEL
Schedule Selection Options		
Specifies the actual start variable	CHART	A_START=
Specifies the actual finish variable	CHART	A_FINISH=
Specifies the baseline start variable	CHART	B_START=
Specifies the baseline finish variable	CHART	B_FINISH=
Concatenates early/late and actual schedules	CHART	COMBINE
Specifies the early start variable	CHART	E_START=
Specifies the early finish variable	CHART	E_FINISH=
Specifies the late start variable	CHART	L_START=
Specifies the late finish variable	CHART	L_FINISH=
Specifies the resource-constrained start variable	CHART	S_START=
Specifies the resource-constrained finish variable	CHART	S_FINISH=

Table 8.1 *continued*

Description	Statement	Option
Timenow Line Options		
Specifies the timenow line color	CHART	CTNOW=
Specifies the timenow line style	CHART	LTNOW=
Suppresses the timenow label	CHART	NOTNLABEL
Specifies the placement of the timenow line	CHART	TIMENOW=
Specifies the timenow line character	CHART	TNCHAR=
Specifies the timenow line width	CHART	WTNOW=
Text Formatting Options		
Specifies the text color	CHART	CTEXT=
Specifies the text font	CHART	FONT=
Specifies the text height multiplier	CHART	HEIGHT=
Specifies the vertical offset for the activity text	CHART	HTOFF=
Web Options		
Specifies the Imagemap output data set	GANTT	IMAGEMAP=
Specifies the HTML variable	CHART	WEB=
Zone Options		
Specifies the zone line color	CHART	CZONE=
Specifies the zone line style	CHART	LZONE=
Suppresses the zone column	CHART	NOZONECOL
Displays only new zone values	CHART	ONEZONEVAL
Specifies the zone line width	CHART	WZONE=
Specifies the zone variable	CHART	ZONE=
Specifies the zone line offset	CHART	ZONEOFF=
Specifies the zone line span	CHART	ZONESPAN=

PROC GANTT Statement

PROC GANTT *options* **;**

The following options can appear in the PROC GANTT statement.

ANNOTATE=*SAS-data-set*

ANNO=*SAS-data-set*

specifies the input data set that contains the appropriate Annotate variables for the purpose of adding text and graphics to the Gantt chart. The data set specified must be an Annotate-type data set. See the section "Annotate Processing" on page 538 for information specifically on annotate processing with the GANTT procedure.

The data set specified with the ANNOTATE= option in the PROC GANTT statement is a "global" ANNOTATE= data set, in the sense that the information in this data set is displayed on every Gantt chart produced in the current invocation of PROC GANTT. This option is available only in graphics mode.

See Example 8.21, "Using the SAS/GRAPH ANNOTATE= Option," for further illustration of this option.

CALEDATA=*SAS-data-set*

CALENDAR=*SAS-data-set*

identifies a SAS data set that specifies the work pattern during a standard week for *each* of the calendars that is to be used in the project. Each observation of this data set (also referred to as the Calendar data set) contains the name or the number of the calendar being defined in that observation, the names of the shifts or work patterns used each day, and, optionally, a standard workday length in hours. For details on the structure of this data set, see the section "Multiple Calendars and Holidays" on page 531. The work shifts referred to in the CALEDATA data set are defined in the WORKDATA data set.

DATA=*SAS-data-set*

names the SAS data set that carries the schedule information to be used by PROC GANTT. If the DATA= option is omitted, the most recently created SAS data set is used. This data set, also known as the Schedule data set, contains all the time variables (early, late, actual, resource-constrained, and baseline start and finish times, and any other variables to be specified in a CHART statement) that are to be plotted on the chart. For projects that use multiple calendars, this data set also identifies the calendar that is used by each activity. The Schedule data set also contains precedence information when drawing a Logic Gantt chart in graphics mode. See the section "Schedule Data Set" on page 526 for more details.

FULLSCREEN

FS

indicates that the Gantt chart be drawn in full-screen mode. This mode enables you to scroll horizontally and vertically through the output using commands, pull-down menus, or function keys. See the section "Full-Screen Version" on page 532 for more information.

GOUT=*graphics catalog*

specifies the name of the graphics catalog used to save the output produced by PROC GANTT for later replay. This option is available only in graphics mode.

GRAPHICS

indicates that the Gantt chart produced be of high-resolution quality. This is the default mode of display. If you invoke the GANTT procedure in Graphics mode, but you do not have SAS/GRAPH software licensed at your site, the procedure stops and issues an error message. See the section "Graphics Version" on page 536 for more information.

HOLIDATA=*SAS-data-set*

names the SAS data set that specifies holidays. These holidays can be associated with specific calendars that are also identified in the HOLIDATA data set (also referred to as the Holiday data set). The HOLIDATA= option must be used with the HOLIDAY= option in the CHART statement, which specifies the variable in the SAS data set that contains the start time of holidays. Optionally, the data set can include a variable that specifies the length of each holiday or a variable that identifies the finish time of each holiday (if the holidays are longer than one unit of the INTERVAL= option). For projects involving multiple calendars, this data set can also include the variable named by the CALID= option that identifies the calendar to be associated with each holiday.

IMAGEMAP=*SAS-data-set*

names the SAS data set that receives a description of the areas of a graph and a link for each area. This information is for the construction of HTML image maps. You use a SAS DATA step to process the output file and generate your own HTML files. The graph areas correspond to the link information that comes from the WEB variable in the schedule data set. This gives you complete control over the appearance and structure of your HTML pages.

LABDATA=*SAS-data-set*

LABELDATA=*SAS-data-set*

LABEL=*SAS-data-set*

specifies the input data set that contains the label specific information. This option is required to initiate the automatic text annotation of the Gantt chart. See the section "Label Data Set" on page 554 for information on the variables it can contain. This option is available only in graphics mode.

LABMAXINT=*n*

LMI=*n*

specifies the maximum number of digits in the integer part when displaying an unformatted numeric as a string. The default value is 16. The maximum number of decimal positions is specified using the MAXDEC= option in the PROC GANTT statement. This option is applicable only to labels defined with the Label data set.

LINEPRINTER

LP

indicates that the Gantt chart be drawn in line-printer mode.

MAXDEC=*n*

M=*n*

indicates the maximum number of decimal positions displayed for a number. A decimal specification in a format overrides a MAXDEC= specification. The default value of MAXDEC= is 2.

PRECDATA=*SAS-data-set*

names the SAS data set that contains the variables that define the precedence constraints in AON format. This data set is required if the Schedule data set does not contain the required precedence information as, for example, when the COLLAPSE option in PROC CPM causes some observations to be excluded from the Schedule data set. When this option is specified, it is mandatory that the ACTIVITY variable exist in both data sets and be identical in both type and length. This option is available only in graphics mode.

SPLIT=*'character'*

S=*'character'*

splits labels used as column headings where the split character appears. When you define the value of the split character, you must enclose it in single quotes. In PROC GANTT, column headings for ID variables consist of either variable labels (if they are present and space permits) or variable names. If the variable label is used as the column heading, then the split character determines where the column heading is to be split.

WORKDATA=*SAS-data-set*

WORKDAY=*SAS-data-set*

identifies a SAS data set that defines the work pattern during a standard working day. Each numeric variable in this data set (also referred to as the Workday data set) is assumed to denote a unique shift pattern during one working day. The variables must be formatted as SAS time values, and the observations are assumed to specify, alternately, the times when consecutive shifts start and end.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC GANTT to obtain separate Gantt charts for observations in groups defined by the BY variables. When a BY statement appears, the procedure expects the schedule data to be sorted in order of the BY variables. If your Schedule data set is not sorted, use the SORT procedure with a similar BY statement to sort the data. The chart for each BY group is formatted separately based only on the observations within that group.

CHART Statement

CHART *specifications / options* **;**

The options that can appear in the CHART statement are listed below. The options are classified under appropriate headings: first, all options that are valid for all modes of the procedure are listed, followed by the options classified according to the mode (line-printer, full-screen, or graphics) of invocation of the procedure. Most of the options in line-printer and full-screen modes are also valid in graphics mode with similar interpretations. The differences and similarities in interpretation of the options are documented under the section "Mode-Specific Differences" on page 558.

General Options

The CHART statement controls the format of the Gantt chart and specifies additional variables (other than early, late, actual, resource-constrained, and baseline start and finish times) to be plotted on the chart. For example, suppose a variable that you want to specify in the CHART statement is one that contains the target finish date for each activity in a project; that is, if FDATE is a variable in the Schedule data set containing the desired finish date for each activity, the CHART statement can be used to mark the value of FDATE on the chart for each activity. A CHART specification can be one of the following types:

variable_1 . . . variable_n

variable_1='symbol_1' . . . variable_n='symbol_n'

(variables)='symbol_1' . . . (variables)='symbol_n'

variable_1 . . . variable_n

indicates that each variable is to be plotted using the default symbol, the first character of the variable name. For example, the following statement

```
CHART SDATE FDATE;
```

causes the values of SDATE to be plotted with an ‘S’ and the values of FDATE with an ‘F.’

variable_1=‘symbol_1’ . . . variable_n=‘symbol_n’
indicates that each variable is to be plotted using the symbol specified. The symbol must be a single character enclosed in quotes.

(variables)=‘symbol_1’ . . . (variables)=‘symbol_n’
indicates that each variable within the parentheses is to be plotted using the symbol associated with that group. The symbol must be a single character enclosed in single quotes. For example, the following statement

```
CHART (ED SD)='*'
      (FD LD)='+';
```

plots the values of the variables in the first group using an asterisk (‘*’) and the values of the variables in the second group using a plus sign (‘+’).

A single CHART statement can contain specifications in more than one of these forms. Also, each CHART statement produces a separate Gantt chart.

NOTE: It is not necessary to specify a CHART statement if default values are to be used to draw the Gantt chart.

The following options can appear in the CHART statement.

A_FINISH=*variable*

AF=*variable*
specifies the variable that contains the actual finish time of each activity in the Schedule data set. This option is not required if the default variable name A_FINISH is used.

A_START=*variable*

AS=*variable*
specifies the variable that contains the actual start time of each activity in the Schedule data set. This option is not required if the default variable name A_START is used.

B_FINISH=*variable*

BF=*variable*
specifies the variable that contains the baseline finish time of each activity in the Schedule data set. This option is not required if the default variable name B_FINISH is used.

B_START=*variable*

BS=*variable*
specifies the variable that contains the baseline start time of each activity in the Schedule data set. This option is not required if the default variable name B_START is used.

BETWEEN=*number*
specifies the number of columns between two consecutive ID variable columns. This option gives you greater flexibility in spacing the ID columns. The default value of the BETWEEN= option is 3.

CALID=*variable*

specifies the variable in the Schedule, Holiday, and Calendar data sets that is used to identify the name or number of the calendar to which each observation refers. This variable can be either numeric or character depending on whether the different calendars are identified by unique numbers or names, respectively. If this variable is not found in any of the three data sets, PROC GANTT looks for a default variable named _CAL_ in that data set (a warning message is issued to the log). For each activity in the Schedule data set, this variable identifies the calendar that is used to mark the appropriate holidays and weekends for the activity. For further details, see the section "Multiple Calendars and Holidays" on page 531.

COMBINE

concatenates the early/late and actual schedule bars of an activity into a single bar and draws a timenow line on the Gantt chart. The COMBINE option does not affect the resource-constrained or baseline schedule bars. If the TIMENOW= option is not specified, it is implicitly assumed to exist and set to missing. The computation of TIMENOW is then carried out as described in the TIMENOW= option. Since the timenow line represents the instant at which a "snapshot" of the project is taken, values less than TIMENOW can be regarded as the "past" and values greater or equal to TIMENOW can be regarded as the "future." The GANTT procedure uses this property of the timenow line to partition the chart into two regions; the region to the left of the timenow line reporting only the actual schedule (events that have already taken place), and the region to the right (including the timenow line) reporting only the predicted early/late schedule.

CRITFLAG

FLAG

indicates that critical jobs be flagged as being critical or super-critical. An activity is critical if its total float is zero. If the total float is negative, the activity is super-critical. Critical activities are marked 'CR,' and super-critical activities are marked 'SC' on the left side of the chart.

DAYLENGTH=*daylength*

specifies the length of the workday. Each workday is plotted starting at the beginning of the day as specified in the DAYSTART= option and ending *daylength* hours later. The value of *daylength* should be a SAS time value. If the INTERVAL= option is specified as DTSECOND, DTMINUTE, DTHOUR, or DTDAY, the default value of *daylength* is 24 hours. If the INTERVAL= option is specified as WORKDAY or DTWRKDAY, the default value of *daylength* is 8 hours. For other values of the INTERVAL= option, the DAYLENGTH= option is ignored.

NOTE: The DAYLENGTH= option is needed to mark the non-worked periods within a day correctly (if the MARKBREAK option is in effect). The DAYLENGTH= option is also used to determine the start and end of a weekend precisely (to the nearest second). This accuracy is needed if you want to depict on a Gantt chart the exact time (for example, to within the nearest hour) for the start and finish of holidays or weekends. This option is used only if the times being plotted are SAS datetime values.

DAYSTART=*daystart*

specifies the start of the workday. The end of the day, *dayend*, is computed as *daylength* seconds after *daystart*. The value of *daystart* should be a SAS time value. This option is to be specified only when the value of the INTERVAL= option is one of the following: WORKDAY, DTSECOND, DTMINUTE, DTHOUR, DTDAY, or DTWRKDAY. For purposes of denoting on the Gantt chart, the weekend is assumed to start at *dayend* on Friday and end at *daystart* on Monday morning. Of course, if the SCALE= and MININTERVAL= values are such that the resolution is not very high, you will be unable to discern the start and end of holidays and weekends to the nearest hour. The default value of *daystart* is 9:00 a.m. if INTERVAL=WORKDAY or INTERVAL=DTWRKDAY, and midnight otherwise.

DUPOK

causes duplicate values of ID variables *not to be skipped*. As described later in the ID Statement section, if two or more consecutive observations have the same combination of values for all the ID variables, only the first of these observations is plotted. The DUPOK option overrides this behavior and causes *all* the observations to be plotted.

DURATION=*variable*

DUR=*variable*

identifies a variable in the Schedule data set that determines whether or not an activity is to be regarded as a milestone with respect to a specific schedule. This option is not required if the default variable name _DUR_ is used. A value of 0 for this variable indicates that if the start and finish times of the activity with respect to a given schedule are identical (a schedule taken to mean early, late, actual, resource-constrained or baseline), then the activity is represented by a milestone with respect to the given schedule. A nonzero value treats identical start and finish times in the default manner by implicitly padding the finish times as specified by the PADDING= option. The milestone symbol is defined by the MILECHAR= option in line-printer and full-screen modes and by the CMILE= , FMILE= , HMILE= , and VMILE= options in graphics mode; these four options represent the color, font, height, and value of the symbol, respectively. See the descriptions of these options for their default values. To illustrate, suppose that the observations for activities **A** and **B** from the Schedule data set are as follows:

ACTIVITY	E_START	E_FINISH	A_START	A_FINISH	_DUR_
A	27JUL04	27JUL04	31JUL04	31JUL04	1
B	31JUL04	31JUL04	01AUG04	02AUG04	0

In this example, the actual schedule for activity **A** begins on '31JUL04' and finishes at the end of the day, as explained in the section "Schedule Data Set" on page 526. PROC GANTT uses the _DUR_ variable to recognize that activity **A** has nonzero duration, pads the finish time by a PADDING= unit, and displays a bar representing one day. In contrast, the value of '0' for _DUR_ in activity **A** alerts PROC GANTT that padding be ignored for any schedule with identical start and finish times. Consequently, the early schedule for activity **B** is represented on the chart by the milestone symbol at '31JUL04.' The actual schedule, however, not having identical start and finish times, is padded as usual and plotted as starting on '01AUG04' and finishing at the end of '02AUG04.'

E_FINISH=*variable*

EF=*variable*

specifies the variable that contains the early finish time of each activity in the Schedule data set. This option is not required if the default variable name E_FINISH is used.

E_START=*variable*

ES=*variable*

specifies the variable that contains the early start time of each activity in the Schedule data set. This option is not required if the default variable name E_START is used.

FILL

causes each page of the Gantt chart to be filled as completely as possible before a new page is started (when the size of the project requires the Gantt chart to be split across several pages). If the FILL option is not specified, the pages are constrained to contain an approximately equal number of activities. The FILL option is not valid in full-screen mode because all of the activities are plotted on one logical page.

HCONNECT

causes a line to be drawn for each activity from the left boundary of the chart to the beginning of the bar for the activity. This feature is particularly useful when the Gantt chart is drawn on a large page. In this case, the schedule bars for some of the activities may not start close enough to the left boundary of the chart; the connecting lines help to identify the activity associated with each bar.

HOLIDAY=(*variable*)

HOLIDAYS=(*variable*)

specifies the date or datetime variable in the Holiday data set that identifies holidays to be marked on the schedule. If there is no end time nor duration specified for the holiday, it is assumed to start at the time specified by the HOLIDAY variable and last one unit of *interval*, where *interval* is the value of the INTERVAL= option.

HOLIDUR=(*variable*)

HDURATION=(*variable*)

specifies the variable in the Holiday data set that identifies the durations of the holidays that are to be marked on the schedule.

HOLIFIN=(*variable*)

HOLIEND=(*variable*)

specifies the date or datetime variable in the Holiday data set that identifies the finish times of the holidays that are to be marked on the schedule.

IDPAGES

displays ID variables on every page. By default, the ID variables are displayed only on the first page.

INCREMENT=*increment*

specifies the number of *minintervals* between time axis labels on the Gantt chart. If the INCREMENT= option is not specified, a value is chosen that provides the maximum possible labeling.

INTERVAL=*interval*

HOLINTERVAL=*interval*

specifies the units for the values of the HOLIDUR variables. Valid values for this option are DAY, WEEKDAY, WORKDAY, DTSECOND, DTMINUTE, DTHOUR, DTDAY, or DTWRKDAY. If the value for the INTERVAL= option has been specified as WEEKDAY, WORKDAY, or DTWRKDAY, weekends are also marked on the Gantt chart with the same symbol as holidays for line-printer quality charts. Graphics-quality Gantt charts use the same PATTERN statement as the one used for marking holidays. The default value of the INTERVAL= option is DAY if the times being plotted are SAS date values and is DTDAY if the times being plotted are SAS datetime values. See the section “Specifying the INTERVAL= Option” on page 532 for further information regarding this option.

L_FINISH=*variable*

LF=*variable*

specifies the variable that contains the late finish time of each activity in the Schedule data set. This option is not required if the default variable name L_FINISH is used.

L_START=*variable*

LS=*variable*

specifies the variable that contains the late start time of each activity in the Schedule data set. This option is not required if the default variable name L_START is used.

MARKBREAK

causes all breaks (non-worked periods) during a day to be marked on the Gantt chart. The symbol used for marking the breaks is the same as the HOLICHAR= symbol. This option may not be of much use unless the chart has been plotted with a scale that enables you to discern the different hours within a day on the Gantt chart. For instance, if the chart is in terms of days, there is no point in trying to show the breaks within a day; on the other hand, if it is in terms of hours or seconds, you may want to see the start and end of the various shifts within a day. This option turns on the MARKWKND option.

MARKWKND

causes all weekends (or non-worked days during a week) to be marked on the Gantt chart. The symbol used for marking weekends is the same as the HOLICHAR= symbol. Note that weekends are also marked on the chart if the value of the INTERVAL= option is WEEKDAY, WORKDAY, or DTWRKDAY.

MAXDATE=*maxdate*

specifies the end time for the time axis of the chart. The default value is the largest value of the times being plotted unless the logic options are invoked without the NOEXTRANGE option in the CHART statement. For a discussion of the default behavior in this instance, see the section "Formatting the Axis" on page 549.

MAXIDS

displays as many consecutive ID variables as possible in the presence of an ID statement. In the absence of this option, the default displays *all* of the variables or *none* if this is not possible.

MINDATE=*mindate*

specifies the starting time for the time axis of the chart. The default value is the smallest value of the times being plotted unless the logic options are invoked without the NOEXTRANGE option in the CHART statement. For a discussion of the default behavior in this instance, see the section "Formatting the Axis" on page 549.

MININTERVAL=*mininterval*

specifies the smallest interval to be identified on the chart. For example, if MININTERVAL=DAY, then one day is represented on the chart by *scale* (see the SCALE= option) number of columns. The default value of the MININTERVAL= option is chosen on the basis of the formats of the times being plotted, as explained in the section "Specifying the MININTERVAL= Option" on page 530. See also the section "Page Format" on page 530 for a further explanation of how to use the MININTERVAL= option in conjunction with the SCALE= option.

NOJOBNUM

suppresses displaying an identifying job number for each activity. By default, the job number is displayed to the left of the Gantt chart.

NOLEGEND

suppresses displaying the concise default legend at the bottom of each page of the Gantt chart. The NOLEGEND option is not effective in full-screen mode.

NOTNLABEL

suppresses displaying the timenow label. By default, the label is displayed on the bottom border of the chart.

PADDING=*padding*

FINPAD=*padding*

requests that finish times on the chart be increased by one *padding* unit. An exception to this is when a milestone is to be plotted. See the DUR= option for further information regarding this. The PADDING= option enables the procedure to mark the finish times as the end of the last time period instead of the beginning. Possible values for *padding* are NONE, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QTR, YEAR, DTSECOND, DTMINUTE, DTHOUR, DTWEEK, DTMONTH, DTQTR, or DTYEAR. The default value is chosen on the basis of the format of the times being plotted. See the section "Specifying the PADDING= Option" on page 529 for further explanation of this option.

PAGELIMIT=*pages*

PAGES=*pages*

specifies an upper limit on the number of pages allowed for the Gantt chart. The default value of *pages* is 100. This option is useful for preventing a voluminous amount of output from being generated by a wrong specification of the MININTERVAL= or SCALE= option. This option is ignored in full-screen mode.

REF=*values*

indicates the position of one or more vertical reference lines on the Gantt chart. The values allowed are constant values. Only those reference lines that fall within the scope of the chart are displayed.

In line-printer and full-screen modes, the reference lines are displayed using the character specified in the REFCHAR= option. In graphics mode, use the CREF=, LREF=, and LWIDTH= options to specify the color, style, and width of the reference lines.

REFLABEL

specifies that the reference lines are to be labeled. The labels are formatted in the same way as the time axis labels and are placed along the bottom border of the Gantt chart at the appropriate points. If the reference lines are too numerous and the scale does not allow all the labels to be nonoverlapping, then some of the labels are dropped.

S_FINISH=*variable*

SF=*variable*

specifies the variable that contains the resource-constrained finish time of each activity in the Schedule data set. This option is not required if the default variable name S_FINISH is used.

S_START=*variable*

SS=*variable*

specifies the variable that contains the resource-constrained start time of each activity in the Schedule data set. This option is not required if the default variable name S_START is used.

SCALE=*scale*

requests that *scale* number of columns on the chart represent one unit of *mininterval* where *mininterval* is the value of the MININTERVAL= option. In line-printer and graphics modes, the default value of the SCALE= option is 1 if the time axis of the chart is too wide to fit on one page. If the time axis fits on less than one page, then a default value is chosen that expands the time axis as much as possible but still fits the time axis on one page. In full-screen mode, the default value of the SCALE= option is always 1.

SKIP=*skip*

S=*skip*

requests that *skip* number of lines be skipped between the plots of the schedules of two activities. The SKIP= option can take integer values between 0 and 4, inclusive. In graphics mode, 0 is not a valid value. The default value of the SKIP= option is 1.

STRIPIDBLANKS

STRIPID

strips all leading blanks from character ID variables. The default behavior is to preserve any leading blanks.

SUMMARY

requests that a detailed description of all symbols and patterns that are used in the Gantt chart be displayed before the first page of the chart. In line-printer mode, this description includes examples of some strings that could occur in the body of the Gantt chart. The SUMMARY option is not supported in full-screen mode.

TIMEAXISFORMAT=*format***|***(format_1 < . . . , format_3 >)*

TAFORMAT=*format***|***(format_1 < . . . , format_3 >)*

specifies formats for up to three rows of time-axis labeling. One time-axis row is displayed for each format specified. The formats control the rows of the time-axis from top to bottom. Missing formats yield a blank row.

TIMENOW=*value*

specifies the position for the timenow line on the chart. If the value is invalid or set to missing, TIMENOW is set to be the time period that follows the maximum of all specified actual times. If there are no actual times, TIMENOW is set to be equal to the current date. The value of TIMENOW is written to the log.

The timenow line has precedence over all other variables and reference lines and is drawn only if it falls within the range of the chart axis. If TIMENOW is based on the maximum of the actual times, and the MAXDATE= option is not specified, then the range of the chart axis is increased, if necessary, to display the timenow line. The timenow line is labeled by default; the label is formatted in the same way as the time axis and is placed along the bottom border of the chart. The timenow line is displayed in line-printer and full-screen modes using the character specified by the TNCHAR= option (or `T`, if none is specified) in the CHART statement. In graphics mode, use the CTNOW=, LTNOW=, and

WTNOW= options in the CHART statement to specify the color, style, and width of the timenow line. In the presence of a timenow line, the actual schedule for an activity with an actual start less than TIMENOW and a missing actual finish time is represented on the Gantt chart by a bar that begins at the actual start and ends at TIMENOW to indicate that the activity is in progress at TIMENOW. This behavior is consistent with the convention used by PROC CPM. A warning is also issued to the log in this case.

USEFORMAT

specifies that the tickmark labels of the Gantt chart axis are to be displayed using the format associated with the first plot variable appearing in the order E_START=, E_FINISH=, L_START=, L_FINISH=, A_START=, A_FINISH=, S_START=, S_FINISH=, B_START=, B_FINISH=. This format is also used for labeling any reference lines and the timenow line.

NOTE: An INFORMAT statement might be necessary to identify user-defined formats. This enables the GANTT procedure to recognize the data type of the start and finish times specified in the input data set.

Full-Screen and Line-Printer Options

The following options can appear in the CHART statement and are specifically for the purpose of producing Gantt charts in line-printer and full-screen modes.

FORMCHAR[*index list* **]**=*'string'*

defines the characters to be used for constructing the chart outlines and dividers. The value is an 11-character string: the first character represents the vertical bar, the second character represents the horizontal bar, and the remaining characters represent the upper left, upper middle, upper right, middle left, middle middle (cross), middle right, lower left, lower middle, and lower right, respectively. The default value of the FORMCHAR= option is `'|----|+|---'`. You can substitute any character or hexadecimal string to customize the chart appearance. Use an index list to specify which default form character each supplied character replaces, or replace the entire default string by specifying the full 11 character replacement string with no index list. For example, change the four corners to asterisks by using

```
formchar(3 5 9 11)= '****'
```

Specifying the following produces charts with no outlines or dividers.

`formchar='           '` (11 blanks)

If you route your output to an IBM 6670 printer that uses an extended font (typestyle 27 or 225) with input character set 216, it is recommended that you specify

```
formchar='FABFACCCBCEB8FECABCBBB'X
```

If you use a printer with a TN (text) print train, it is recommended that you specify the following:

```
formchar='4FBFACBFBC4F8F4FABBFBB'X
```

HCONCHAR=*'character'*

specifies the symbol to be used for drawing the connecting line described in the HCONNECT option. The default character is `-` . This is a line-printer option and is not valid in conjunction with the GRAPHICS option. For corresponding graphics options, see the LHCON= and CHCON= options described later in this section under "Graphics Options."

HOLICHAR=*'character'*

indicates the character to display for holidays. Note that PROC GANTT displays only those holidays that fall within the duration or the slack time of an activity. The default character used for representing holidays is `!` .

JOINCHAR=*'string'*

defines a string eight characters long that identifies nonblank characters to be used for drawing the schedule. The first two symbols are used to plot the schedule of an activity with positive total float. The first symbol denotes the duration of such an activity while the second symbol denotes the slack present in the activity's schedule. The third symbol is used to plot the duration of a *critical* activity (with zero total float). The next two symbols are used to plot the schedule of a *supercritical* activity (one with negative float). Thus, the fourth symbol is used to plot the negative slack of such an activity starting from the late start time (to early start time), and the fifth symbol is used to plot the duration of the activity (from early start to early finish). The sixth symbol is used to plot the actual schedule of an activity if the A_START and A_FINISH variables are specified. The seventh symbol is used to plot the resource-constrained schedule of an activity if the S_START and S_FINISH variables are specified. The eighth symbol is used to plot the baseline schedule of an activity if the B_START and B_FINISH variables are specified. The default value of the JOINCHAR= option is `'-.=-*-*_'` .

MILECHAR=*'character'*

indicates the character to display for the milestone symbol. If this option is not used, the letter `M` is used. In the event that another milestone or a character representing a start or finish time is to be plotted in this column, the OVERLAPCH= character is used.

OVERLAPCH=*'character'*

OVLPCHAR=*'character'*

indicates the overprint character to be displayed when more than one of the symbols in SYMCHAR=*'string'* or MILECHAR=*'character'* are to be plotted in the same column. The default character is `*` .

OVPCHAR=*'character'*

indicates the character to be displayed if one of the variables specified in the CHART statement is to be plotted in the same column as one of the start or finish times. If no OVPCHAR= option is given, the 'at' symbol (`@`) is used. Note that if one of the E_START, E_FINISH, L_START, L_FINISH, A_START, A_FINISH, S_START, S_FINISH, B_START, or B_FINISH times coincides with another, the overprint character to be displayed can be specified separately using the OVERLAPCH= option.

REFCHAR=*'character'*

indicates the character to display for reference lines. If no REFCHAR= option is given, the vertical bar (|) is used. If a time variable value is to be displayed in the column where a REF= value goes, the plotting symbol for the time variable is displayed instead of the REFCHAR= value. Similarly, the HOLICHAR= symbol has precedence over the REFCHAR= value.

SYMCHAR=*'string'*

defines the symbols to be used for plotting the early start, late start, early finish, late finish, actual start and finish, resource-constrained start and finish, and baseline start and finish times, in that order. The default value is `'<<>>**<>[]'`. If any of the preceding symbols coincide with one another or with the milestone symbol, the symbol plotted is the one specified in the OVERLAPCH= option (or `*`, if none is specified). If the actual, resource-constrained, and baseline schedules are not plotted on the chart, you can specify only the first four symbols. If fewer than the required number of symbols are specified, nonspecified symbols are obtained from the default string.

TNCHAR=*'character'*

indicates the character to display for the timenow line. If this option is not used, the letter `T` is used.

Graphics Options

The following describes the interpretation of the CHART specification in graphics mode. Note that the GANTT procedure is not supported with the ActiveX or Java series of devices on the GOPTIONS statement.

As before, the CHART statement controls the format of the Gantt chart and specifies additional variables (other than the early, late, actual, resource-constrained, and baseline start and finish times) to be plotted on the chart. The same forms for the specification of CHART variables (as in the line-printer and full-screen version) are allowed, although the interpretation is somewhat different. Each form of specification is repeated here with a corresponding description of the interpretation. Note that the symbols for any activity are plotted on a line above the one corresponding to that activity. In addition to plotting the required symbol, PROC GANTT draws a vertical line below the symbol in the same color as the symbol. The length of the line is the same as the height of the bars (referred to as bar height) that represent the durations of the activities on the Gantt chart. This line helps identify the exact position of the plotted value. See also the section "Special Fonts for Project Management and Decision Analysis" on page 544 for information on a special set of symbols that are suitable for representing CHART variables on a Gantt chart.

variable1 . . . variablen

indicates that each variable is to be plotted using symbols specified in SYMBOL statements. The *i*th variable in the list is plotted using the plot symbol, color, and font specified in the *i*th SYMBOL statement. The height specified in the SYMBOL statement is multiplied by the bar height to obtain the height of the symbol that is plotted. Thus, if `H=0.5` in the first SYMBOL statement, and the bar height is 5% of the screen area, then the first symbol is plotted with a height of 2.5%. For example, suppose the following two SYMBOL statements are in effect:

```
SYMBOL1 V=STAR C=RED   H=1;
SYMBOL2 V=V    C=GREEN H=0.5 F=GREEK;
```

Then, the following statement

```
CHART SDATE FDATE;
```

causes values of SDATE to be plotted with a red star that is as high as each bar and the values of FDATE with an inverted green triangle that is half as high as the bar height. See the section "Using SYMBOL Statements" on page 542 for further information on using the SYMBOL statement.

variable1='symbol1' . . . variablen='symboln'
indicates that each variable is to be plotted using the symbol specified. The symbol must be a single character enclosed in quotes. The font used for the symbol is the same as the font used for the text.

(variables)='symbol1' . . . (variables)='symboln'
indicates that each variable in parentheses is to be plotted using the symbol associated with that group. The symbol must be a single character enclosed in single quotes. For example, the following statement

```
CHART (ED SD)='*'
      (FD LD)='+';
```

plots the values of variables in the first group using an asterisk (*) and the values of variables in the second group using a plus sign (+).

A single CHART statement can contain requests in more than one of these forms.

NOTE: It is not necessary to specify a CHART statement if only default values are used to draw the Gantt chart.

The following options can appear in the CHART statement specifically for the production of high-resolution graphics quality Gantt charts.

ABARHT=*h*
specifies that the height of the actual schedule bar be *h* cellheights. The value of *h* is restricted to be a positive real number. The default bar height is one cellheight. This specification will override a BARHT= specification. In the event that the actual schedule bar corresponds to the logic bar (using the LEVEL= option), the value is ignored and the default value is used instead. Any non-working days corresponding to this schedule bar are also drawn using the same height as the schedule bar unless the HBARHT= option is specified.

ABAROFF=*d*
specifies that the actual schedule bar be offset *d* cellheights from its default position. A value of zero corresponds to the default position. The direction of increase is from top to bottom. This specification will override a BAROFF= specification. In the event that the actual schedule bar corresponds to the logic bar (specified using the LEVEL= option), the value is ignored and the default value is used instead. Any non-working days corresponding to this schedule bar are drawn using the offset of the schedule bar unless the HBAROFF= option is specified.

ACTIVITY=*variable*

ACT=*variable*
specifies the variable identifying the names of the nodes representing activities in the Schedule data set. This option is required when the precedence information is specified using the AON format. The variable can be either numeric or character in type. If the PRECDATA= option is specified, then this variable must also exist in the Precedence data set and have identical type and length.

ANNOTATE=*SAS-data-set*

ANNO=*SAS-data-set*
specifies the input data set that contains the appropriate Annotate variables for the purpose of adding text and graphics to the Gantt chart. The data set specified must be an Annotate-type data set. See also the section "Annotate Processing" on page 538 for information specifically on annotate processing with the GANTT procedure.

The ANNOTATE= data set specified in a CHART statement is used only for the Gantt chart created by that particular CHART statement. You can also specify an ANNOTATE= data set in the PROC GANTT statement, which provides "global" Annotate information to be used for all Gantt charts created by the procedure.

AOA

causes PROC GANTT to use the specification for the AOA format for producing a Logic Gantt chart when the precedence information has been specified in both AOA format (TAIL= and HEAD= options) and AON format (ACTIVITY=, SUCCESSOR=, and, optionally, LAG= options). The default behavior is to use the AON format.

BARHT=*h*

specifies that the height of all the schedule bars be *h* cellheights. The value of *h* is restricted to be a positive real number. The default value is one cellheight. This specification can be overridden for each schedule type by specifying the bar height option appropriate for that schedule type. If a Logic Gantt chart is produced, the specified bar height is ignored for the logic bar (specified using the LEVEL= option) and the default bar height of one cellheight is used for it instead. All non-working days corresponding to a schedule bar are drawn using the height of the schedule bar unless the HBARHT= option is specified.

BAROFF=*d*

specifies that all the schedule bars be offset *d* cellheights from their default positions. A value of zero corresponds to the default positions. The direction of increase is from top to bottom. This specification can be overridden for each schedule type by specifying the bar offset option that is appropriate for that schedule type. If a Logic Gantt chart is produced, the specified bar offset is ignored for the logic bar (specified using the LEVEL= option) and the default bar offset of zero used instead.

BBARHT=*h*

specifies that the height of the baseline schedule bar be *h* cellheights. The value of *h* is restricted to be a positive real number. The default bar height is one cellheight. This specification overrides a BARHT= specification. In the event that the baseline schedule bar corresponds to the logic bar (using the LEVEL= option), the value is ignored and the default value is used instead. Any non-working days corresponding to this schedule bar are also drawn using the same height as the schedule bar unless the HBARHT= option is specified.

BBAROFF=*d*

specifies that the baseline schedule bar be offset *d* cellheights from its default position. A value of zero corresponds to the default position. The direction of increase is from top to bottom. This specification overrides a BAROFF= specification. In the event that the baseline schedule bar corresponds to the logic bar (specified using the LEVEL= option), the value is ignored and the default value is used instead. Any non-working days corresponding to this schedule bar are drawn using the offset of the schedule bar unless the HBAROFF= option is specified.

BOTTOM

BJUST

positions the bottom of the Gantt chart at the bottom of the page, just above the footnotes. This option is ignored if you specify the TOP or TJUST option.

CAXIS=*color*

CAXES=*color*

CA=*color*

specifies the color to use for displaying axes for the Gantt chart. The default color depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 562 for more information.

CFRAME=*color*

CFR=*color*

specifies the color to use for filling the axis area. If the CFRAME= option is not specified and the GSTYLE system option is not in effect, then the axis area is not filled. If the GSTYLE system option is in effect, then the default color depends on the current ODS style template; see the section "ODS Style Templates" on page 562 for more information. The CFRAME= option is ignored if the NOFRAME option is specified.

CHARTWIDTH=*p*

CHARTPCT=*p*

specifies the width of the axis area as a percentage of the total Gantt chart width in the chart that would be produced if you had a page large enough to contain the entire chart without compression. The Gantt procedure rescales the chart so the axis area with is *p*% of the virtual chart width and the text area width is (100-*p*)% of the virtual chart width.

This option gives you the capability to generate Gantt charts that are consistent in their appearance. In the event that the chart fits on a single page, it is possible to get a smaller chart than had the CHARTWIDTH= option not been specified. You can use the FILL option in this case if you wish to use the entire page.

CHCON=*color*

specifies the color to use for drawing the horizontal connecting lines. The default color depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 562 for more information.

CMILE=*color*

specifies the color to use for drawing the milestone symbol on the chart. If the CMILE= option is not specified, the default color of the milestones follows the rules for coloring the bars of the relevant schedule. For example, the milestone depicting a critical activity is drawn with the color of the fill pattern used for critical activities. For an activity with slack, the early start and late start milestone are drawn with the color of the fill pattern used for the duration and the slack time of a noncritical activity, respectively. You can also control the color at the activity level by using a PATTERN variable.

COMPRESS

specifies that the Gantt chart be drawn on the number of output pages determined by the HPAGES= and VPAGES= options. If the HPAGES= option is not specified, the procedure assumes a default of HPAGES=1. If the VPAGES= option is not specified, the procedure assumes a default of VPAGES=1. The COMPRESS option does not attempt to maintain the aspect ratio of the Gantt chart. To maintain the aspect ratio of the Gantt chart, use the PCOMPRESS option instead.

CPREC=*color*

specifies the color to use for drawing the precedence connections. The default color depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 562 for more information.

CREF=*color*

specifies the color to use for drawing vertical reference lines on the chart. The default color depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 562 for more information.

CTEXT=*color*

CT=*color*

specifies the color to use for displaying text that appears on the chart, including variable names or labels, tickmark values, values of ID variables, and so on. The default color depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 562 for more information.

CTEXTCOLS=*name*

CTEXTCOLS=*(namelist)*

CPATTEXT=*name*

CPATTEXT=*(namelist)*

CACTTEXT=*name*

CACTTEXT=*(namelist)*

names the columns of activity text to be displayed using the color of the PATTERN variable when one exists or from the fill pattern from a particular schedule bar.

A missing value for a PATTERN variable results in the default text color being used. The default text color is the value of the CTEXT= option.

In the absence of a PATTERN variable, the activity text color is the color of the fill pattern indicating the duration of the schedule identified by the PATLEVEL= option. If PATLEVEL=EARLY or PATLEVEL=LATE, the color depends on the status of the activity. Colors for critical duration, supercritical duration, and normal duration are used depending on whether the activity is critical, supercritical, or noncritical, respectively. If more than one level is specified, the first in order of appearance on the Gantt chart is used, that is, in order EARLY, LATE, ACTUAL, RESOURCE, BASELINE.

Possible values for the CTEXTCOLS= option are shown in the following table.

Value	Interpretation
ZONE	ZONE variable column
JOBNUM	Job number column
ID	ID variable columns
FLAG	Status flag column
ALL	All of the above (default)

CTNOW=*color*

specifies the color to use for drawing the timenow line on the chart. The default color depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 562 for more information.

CZONE=*color*

CZLINE=*color*

specifies the color to use for drawing the horizontal zone lines that demarcate the different zones on the chart. The default color depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 562 for more information.

DESCRIPTION=*'string'*

DES=*'string'*

specifies a descriptive string, up to 40 characters in length, that appears in the description field of the master menu of PROC GREPLAY. If the DESCRIPTION= option is omitted, the description field contains a description assigned by PROC GANTT.

EBARHT=*h*

LBARHT=*h*

specifies that the height of the early/late schedule bar be *h* cellheights. The value of *h* is restricted to be a positive real number. The default bar height is one cellheight. This specification overrides a BARHT= specification. In the event that the early/late schedule bar corresponds to the logic bar (using the LEVEL= option), the value is ignored and the default value is used instead. Any non-working days corresponding to this schedule bar are also drawn using the same height as the schedule bar unless the HBARHT=option is specified.

EBAROFF=*d*

LBAROFF=*d*

specifies that the early/late schedule bar be offset *d* cellheights from its default position. A value of zero corresponds to the default position. The direction of increase is from top to bottom. This specification overrides a BAROFF= specification. In the event that the early/late schedule bar corresponds to the logic bar (specified using the LEVEL= option), the value is ignored and the default value is used instead. Any non-working days corresponding to this schedule bar are drawn using the offset of the schedule bar unless the HBAROFF= option is specified.

FONT=*font*

specifies the font to use for displaying job numbers, ID variables, legend, labels on the time axis, and so forth. The default font depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 562 for more information.

FMILE=*font*

specifies the font to use for drawing the milestone symbol on the chart. To select a symbol from the special symbol table, set FMILE=NONE or leave it unspecified. If the FMILE= option is specified without a corresponding VMILE= option, the value of the FMILE= option is ignored, and the default milestone symbol, a filled diamond, is used instead. A warning is issued to the log in this instance.

See also the section "Special Fonts for Project Management and Decision Analysis" on page 544 for information on a special set of symbols that are suitable for representing milestones on a Gantt chart.

HBARHT=*h*

specifies that all non-working days be displayed with a bar which is *h* cellheights high. The default behavior is to use the same height as that of the schedule bar.

HBAROFF=*d*

specifies that the bars which represent non-working days be offset *d* cellheights from their default positions. The default behavior is to use the same offset as that of the schedule bar.

HEAD=*variable*

HEADNODE=*variable*

specifies the variable (either character or numeric) in the Schedule data set that contains the name of the node that represents the finish of the activity. This option is required when the precedence information is specified using the AOA format.

HEIGHT=*h*

specifies that the height for all text in PROC GANTT, excluding TITLE and FOOTNOTE statements, be *h* times the value of HTEXT=, the default text height specified in the GOPTIONS statement of SAS/GRAPH. The value of *h* is a positive real number; the default value is 1.0.

To illustrate, suppose you have the specification HEIGHT=0.6 in the CHART statement and the following GOPTIONS statement:

```
GOPTIONS htext = 2 in;
```

Then the height for all text in PROC GANTT is 0.6 x 2 in = 1.2 in.

If the value of HTEXT= is not specified in a GOPTIONS statement, then the text height is determined by the font size attribute of the GraphDataText element of the current ODS style template. See the section "ODS Style Templates" on page 562 for more information about ODS styles.

For each activity, all text corresponding to the JOB, FLAG, and ID variables is displayed at a depth of *d* cells from the top of the first bar corresponding to the activity, where *d* is the value of the HTOFF= option. The default value of *d* is 1.0. Furthermore, the text strings do not overwrite one another and *skip*, the value of the SKIP= option, is not increased to accommodate a large text height. Subject to the preceding restrictions, PROC GANTT calculates the maximum allowable value for text height as the height occupied by (*skip* + *the number of different schedule bars drawn per activity*) blank lines. Specifically, this is the height between like bars corresponding to consecutive activities. If the specified text height exceeds this value, the height is truncated to the maximum allowable value and a warning is issued to the log. This option enables you to enlarge the text to at least the height occupied by all of the schedule bars, making it easier to read. This is especially useful when the value of the VPOS= option is very large, and several schedule bars are plotted for each activity. It also provides easier identification of the activity corresponding to a given schedule bar.

HMILE=*height*

specifies the height in cells of the milestone symbol. The height is a positive real number; the default value is 1.0.

HPAGES=*h*

specifies that the Gantt chart is to be produced using *h* horizontal pages. This, however, may not be possible due to intrinsic constraints on the output. For example, the GANTT procedure requires that every horizontal page represent at least one activity. Thus, the number of horizontal pages can never exceed the number of activities in the project. Subject to such inherent constraints, the GANTT procedure attempts to use the specified value for the HPAGES= option; if this fails, it uses *h* as an upper bound. The exact number of horizontal pages used by the Gantt chart is given in the _ORGANTT macro variable. See the section "Macro Variable _ORGANTT" on page 560 for further details.

The appearance of the chart with respect to the HPAGES= option is also influenced by the presence of other related procedure options. The HPAGES= option performs the task of determining the number of vertical pages in the absence of the VPAGES= option. If the COMPRESS or PCOMPRESS option is specified in this scenario, the chart uses one vertical page; if neither option is specified, the number of vertical pages is computed to display as much of the chart as possible in a proportional manner.

HTOFF=*d*

specifies that the line upon which all activity text rests, also referred to as the *font baseline*, is positioned at a depth of *d* cells below the top of the first bar. The default value of *d* is 1.0. The value of the HTOFF= option can be any nonnegative real number less than the (*skip* + *the number of different schedule bars per activity* - 1). A value of 0 positions text on the line corresponding to the top of the first bar. Assigning the maximum value corresponds to positioning text directly above the bar reserved for CHART variables of the next activity on the page. If a value larger than the maximum is specified, PROC GANTT truncates this value to the maximum and issues a warning to the log. Furthermore, if the HEIGHT= and HTOFF= values cause activity text to overwrite the text headings, PROC GANTT reduces the HTOFF= value accordingly and issues a warning to the log.

LABVAR=*variable*

specifies the variable that links observations in the Label data set (label definitions) to observations in the Schedule data set (activities). This variable must exist in both the Schedule data set and the Label data set and be identical in type and length. The variable can be either numeric or character in type. The linking can be a 1-1, 1-many, many-1, or many-many relationship. The linking can be used to extract positional information as well as the text string information from the Schedule data set for an observation in the Label data set when such information cannot be retrieved from the relevant variables in the Label data set.

If the _Y variable does not exist or its value is missing, the vertical coordinate for a label's placement position is determined from the activities that are linked to it and their relative positions on the activity axis of the Gantt chart. A value of -1 for _Y implies linking of the label to every activity (assuming data values are used). This is equivalent to specifying the LABVAR= option in the CHART statement and linking every activity to the label. Note that any Label data set observation with dual linkage definitions is ignored. That is, an observation with _Y equal to -1 and with a nonmissing value for the LABVAR= variable is ignored.

The following rules apply to label definitions in the Label data set that are linked to activities in the Schedule data set:

- If the _X variable does not exist in the Label data set or its value is missing, the horizontal coordinate is extracted from the Schedule data set using the _XVAR variable.
- If the _LABEL variable does not exist in the Label data set or its value is missing, the text string is determined from the Schedule data set using the _LVAR variable.

LABRULE=*rule*

LABFMT=*rule*

specifies the rule to use for laying out labels that are defined in the Label data set. Valid values for *rule* are PAGECLIP and FRAMCLIP. PAGECLIP displays a label at the specified location and clips any part of the label that runs off the page. A value of FRAMCLIP differs from PAGECLIP in that it clips all labels with data value coordinates that run off the frame of the Gantt chart. The default value for *rule* is PAGECLIP.

LABSPLIT=*'character'*

LABELSPLIT=*'character'*

splits labels that are defined in the Label data set wherever the split character appears. By default, if there are embedded blanks, the GANTT procedure attempts to split strings at suitable blanks so that the resulting lines are equal in length. To suppress the default splitting when using strings embedded with blanks, specify a dummy character not used in the labeling.

LAG=*variable*

LAG=(*variables***)**

specifies the variables identifying the lag types of the precedence relationships between an activity and its successors. Each SUCCESSOR variable is matched with the corresponding LAG variable; that is, for a given observation, the *i*th LAG variable defines the relationship between the activities specified by the ACTIVITY variable and the *i*th SUCCESSOR variable. The LAG variables must be character type and their values are expected to be specified as one of FS, SS, SF, FF, which denote 'Finish-to-Start', 'Start-to-Start', 'Start-to-Finish', 'Finish-to-Finish', respectively. You can also use the *keyword_duration_calendar* specification used by PROC CPM although PROC GANTT uses only the *keyword* information and ignores the lag *duration* and the lag *calendar*. If no LAG variables exist or if an unrecognized value is specified for a LAG variable, PROC GANTT interprets the lag as a 'Finish-to-Start' type. If the PRECDATA= option is specified, the LAG variables are assumed to exist in the Precedence data set; otherwise, they are assumed to exist in the Schedule data set.

LEFT

LJUST

displays the Gantt chart left-justified with the left edge of the page. This option has priority over the RIGHT or RJUST option. Note that when displaying a Gantt chart in graphics mode, the chart is centered in both horizontal and vertical directions in the space available after accounting for titles, footnotes, and notes. The chart justification feature enables you to justify the chart in the horizontal and vertical directions with the page boundaries.

LEVEL=*number*

specifies the schedule bar to use for drawing the precedence connections. The default value of LEVEL= is 1, which corresponds to the topmost bar.

LHCON=*linetype*

specifies the line style (1 – 46) to be used for drawing the horizontal connecting line produced by the HCONNECT option described earlier in this section. Possible values for *linetype* are

1	solid line (the default value when LHCON= is omitted)
2 – 46	various dashed lines. See Figure 8.5.

For the corresponding line-printer option, see the HCONCHAR= option described earlier in this section.

LPREC=*linetype*

specifies the line style (1 – 46) to use for drawing the precedence connections. The default line style is 1, a solid line. See Figure 8.5 for examples of the various line styles available.

LREF=*linetype*

specifies the line style (1 – 46) to use for drawing the reference lines. The default line style is 1, a solid line. See Figure 8.5 for examples of the various line styles available. For the corresponding line-printer option, see the REFCHAR= option described earlier.

LTNOW=*linetype*

specifies the line style (1 – 46) to use for drawing the timenow line. The default line style is 1, a solid line. See Figure 8.5 for examples of the various line styles available.

LWIDTH=*linewidth*

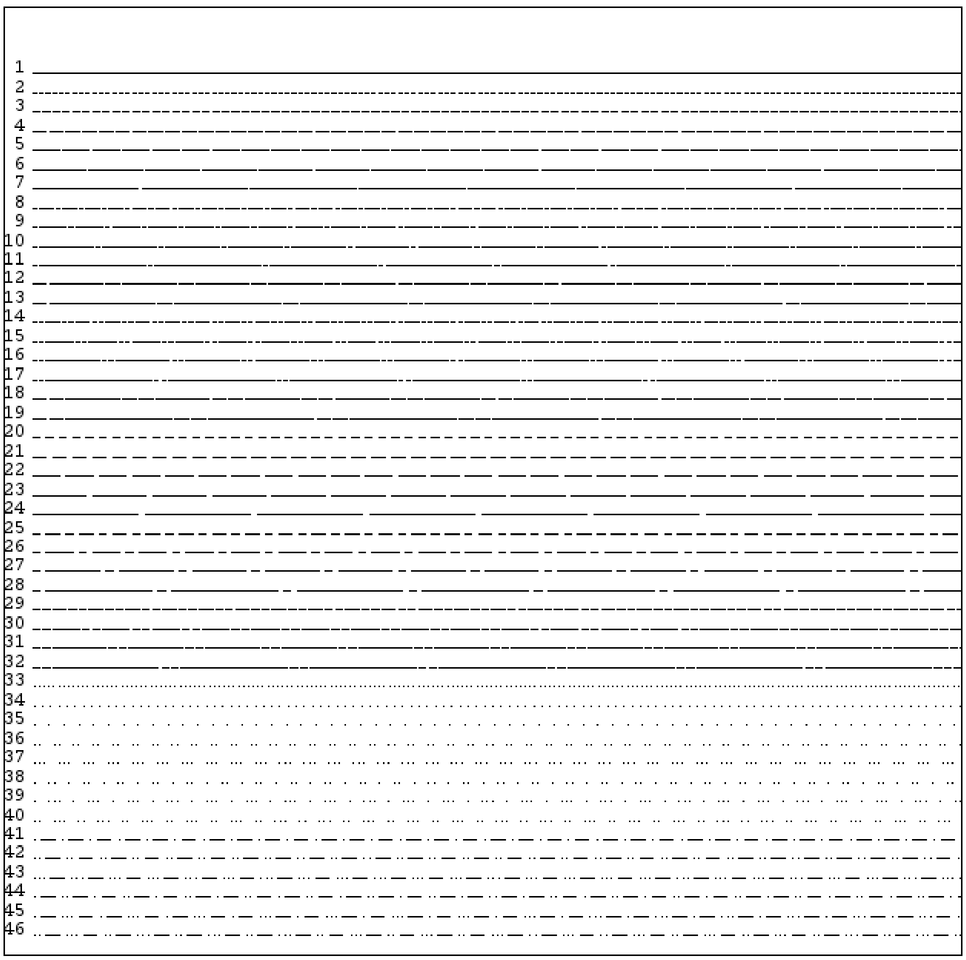
specifies the line width to be used for drawing lines, other than the timenow line and precedence connection lines, used in the Gantt chart. The default width is 1.

LZONE=*linetype*

LZLINE=*linetype*

specifies the line style (1 – 46) to use for drawing the horizontal zone lines which demarcate the different zones on the chart. The default line style is 1, a solid line.

Figure 8.5 Valid Line Styles



MAXDISLV=*columns*

specifies the maximum allowable distance, in number of columns, that a local vertical can be positioned from its minimum offset to avoid overlap with a global vertical. The value of the MAXDISLV= option must be greater than or equal to 0.1; the default value is 1. For the definitions of global and local verticals, see the section "Specifying the Logic Options" on page 546.

MININTGV=*columns*
specifies the minimum inter-distance, in number of columns, of any two global verticals to prevent overlap. The value of the MININTGV= option must be greater than or equal to 0.1; the default value is 0.75.

MINOFFGV=*columns*
specifies the minimum offset, in number of columns, of a global vertical from the end of the bar with which it is associated. The value of the MINOFFGV= option must be greater than or equal to 0.1; the default value is 1.

MINOFFLV=*columns*
specifies the minimum offset, in number of columns, of a local vertical from the end of the bar with which it is associated. The value of the MINOFFLV= option must be greater than or equal to 0.1; the default value is 1.

NAME=*'string'*
where *'string'* specifies a descriptive string, up to eight characters long, that appears in the name field of the master menu of the GREPLAY procedure. If you omit the NAME= option, the name field of the master menu contains the name of the procedure.

NJOBS=*number*

NACTS=*number*
specifies the number of jobs that should be displayed on a single page. This option overrides the VPAGES= option.

NOARROWHEAD

NOARRHD
suppresses the arrowhead when drawing the precedence connections.

NOEXTRANGE

NOXTRNG
suppresses the automatic extension of the chart axis range when drawing a Logic Gantt chart and neither the MINDATE= nor MAXDATE= option is specified.

NOFRAME

NOFR
suppresses drawing the vertical boundaries to the left and right of the Gantt chart; only the top axis and a parallel line at the bottom are drawn. If this option is not specified, the entire chart area is framed.

NOPAGENUM
suppresses numbering the pages of a multipage Gantt chart. This is the default behavior. To number the pages of a multipage Gantt chart on the upper right-hand corner of each page, use the PAGENUM option.

NOPATBAR
suppresses the use of the PATTERN variable for filling the schedule bars. The default fill patterns are used instead. Typically, this option is used when you want to color the activity text using the CTEXTCOLS= option but leave the bars unaffected by the PATTERN variable.

NOTMTIME

suppresses the display of the time portion of the axis tickmark label when the value of MININTERVAL is DTDAY. When MININTERVAL=DTDAY, the time axis tickmarks are labeled with three lines, the first indicating the month, the second indicating the day, and the third indicating the time. This option effectively lowers the first two lines by a line and drops the third line altogether.

NOZONECOL

suppresses displaying the ZONE variable column that is automatically done in the presence of a zone variable.

NTICKS=*number*

NINCRS=*number*

specifies the number of tickmarks that should be displayed on the first horizontal page of the Gantt chart. The number of tickmarks on the remaining horizontal pages is determined by the page width and the columns of text that are to be displayed (ZONE, IDs, flag, and so forth). The page width is determined to be the minimum width necessary to fit the first page. If the IDPAGES option is specified, the number of tickmarks is the same as that specified by the NTICKS= option. This option overrides the HPAGES= option.

ONEZONEVAL

displays the value of the ZONE variable in the ZONE variable column only for activities that begin a new zone. A blank string is displayed for all other activities.

PAGENUM

numbers the pages of the Gantt chart on the top right-hand corner of the page if the chart exceeds one page. The numbering scheme is from left to right, top to bottom.

PATLEVEL=*name*

PATLEVEL=*(namelist)*

specifies the different schedule bar levels to fill using the PATTERN variable. By default, all of the schedule bar levels for an activity are filled using the pattern defined by the PATTERN variable. Note that holiday and non-working days are not filled with this pattern.

Possible values for the PATLEVEL= option and their actions are shown in the following table.

Value	Interpretation
EARLY	Early/Late schedule durations
LATE	Early/Late schedule durations
ACTUAL	Actual schedule durations
RESOURCE	Resource schedule duration
BASELINE	Baseline schedule duration
ALL	All of the above (default)

In the absence of a PATTERN variable, this option defines the schedule type that determines the color for the activity text columns (ZONE variable, ID variable, Job number, Critical Flag), which are identified with the CTEXTCOLS= option. In this case, only one schedule type is used, namely the first one appearing in the order EARLY, LATE, ACTUAL, RESOURCE, BASELINE.

PATTERN=*variable*

PATVAR=*variable*

specifies an integer variable in the Schedule data set that identifies the pattern for filling the schedule bars and coloring the milestones. The default PATTERN variable name is _PATTERN. If the value of the PATTERN variable is missing for a particular activity, or if there is no PATTERN variable, the different schedule bars and milestones for the activity are drawn using the corresponding default patterns given in Table 8.7. The procedure uses the defined or default pattern to fill all the schedule bars and color all the milestones associated with the activity, except for holidays and non-working days. Use the PATLEVEL= option to restrict the application of the defined pattern to selected schedule bar levels.

When plotting split activities, you have the additional capability of overriding the defined pattern at the segment level by specifying a value for the PATTERN variable for the schedule data set observation representing the segment. Setting it to missing results in inheriting the PATTERN variable value from the observation for the same activity with a missing SEGMT_NO. For example, setting PATTERN=SEGMT_NO in the CHART statement when using split activities results in each segment using a different pattern.

Note that, if the value of the PATTERN variable is *n* for a particular activity, the GANTT procedure uses the specifications in the *n*th generated PATTERN definition, not the specifications in the PATTERN *n* statement.

The chart legend and summary, when displayed, indicate the default patterns that identify the different schedule types represented on the Gantt chart as listed in Table 8.7. Since the PATTERN variable overrides these values at the activity level, you must be careful in interpreting the summary and legend when using a PATTERN variable, especially if any of the specified pattern definitions overlap with one of the default patterns.

PCOMPRESS

specifies that every output page of the Gantt chart is to be produced maintaining the original aspect ratio of the Gantt chart. The number of output pages is determined by the HPAGES= and VPAGES= options. In the absence of the HPAGES= and VPAGES= options, the PCOMPRESS option displays the Gantt chart on a single page.

RBARHT=*h*

SBARHT=*h*

specifies that the height of the resource-constrained schedule bar be *h* cellheights. The value of *h* is restricted to be a positive real number. The default bar height is one cellheight. This specification overrides a BARHT= specification. In the event that the resource-constrained schedule bar corresponds to the logic bar (using the LEVEL= option), the value is ignored and the default value is used instead. Any non-working days corresponding to this schedule bar are also drawn using the same height as the schedule bar unless the HBARHT= option is specified.

RBAROFF=*d*

SBAROFF=*d*

specifies that the resource-constrained schedule bar be offset *d* cellheights from its default position. A value of zero corresponds to the default position. The direction of increase is from top to bottom. This specification overrides a BAROFF= specification. In the event that the resource-constrained schedule bar corresponds to the logic bar (specified using the LEVEL= option), the value is ignored and the default value is used instead. Any non-working days corresponding to this schedule bar are drawn using the offset of the schedule bar unless the HBAROFF= option is specified.

RIGHT

RJUST

displays the Gantt chart right-justified with the right edge of the page. This option is ignored in the presence of the LEFT or LJUST option.

SHOWPREC

causes PROC GANTT to terminate in the event that a valid AOA or AON specification exists, and an error occurs either in the logic system (memory allocation, data structure creation, and so on) or simply due to bad data (missing values for the ACTIVITY, TAIL, HEAD variables, and so on). The default behavior is to attempt drawing the chart without the precedence connections.

SUCCESSOR=*variable*

SUCC=*variable*

SUCCESSOR=(*variables***)**

SUCC=(*variables***)**

specifies the variables identifying the names of the immediate successors of the node specified by the ACTIVITY variable. This option is required when the precedence information is specified in the AON format. These variables must have the same type as the ACTIVITY variable. If the PRECDATA= option has been specified, the SUCCESSOR variables are assumed to exist in the Precedence data set; otherwise, they are assumed to exist in the Schedule data set.

TAIL=*variable*

TAILNODE=*variable*

specifies the variable in the Schedule data set that contains the name of the node that represents the start of the activity. This option is required when the precedence information is specified using the AOA format. The variable can be either numeric or character in type.

TOP

TJUST

positions the top of the Gantt chart at the top of the page, just below the titles. This option has priority over the BOTTOM or BJUST option.

VMILE=*value*

specifies a plot symbol from the font specified in the FMILE= option to be used as the milestone symbol on the chart. If the FMILE= option is set to NONE or is not specified, then the milestone symbol is the symbol specified by the VMILE= option in the special symbol table shown in Figure 8.7. The default milestone symbol is a filled diamond.

VPAGES=*v*

Specifies that the Gantt chart is to be produced using *v* vertical pages. This, however, may not be possible due to intrinsic constraints on the output. For example, the GANTT procedure requires that every vertical page represent at least one tickmark. Thus, the number of vertical pages can never exceed the number of tickmarks in the axis. Subject to such inherent constraints, the GANTT procedure attempts to use the specified value for the VPAGES= option; if this fails, it uses *v* as an upper bound. The exact number of vertical pages used by the Gantt chart is provided in the _ORGANTT macro variable. See the section "Macro Variable _ORGANTT" on page 560 for further details.

The appearance of the chart with respect to the VPAGES= option is also influenced by the presence of other related procedure options. The VPAGES= option performs the task of determining the number of horizontal pages in the absence of the HPAGES= option. If the COMPRESS or PCOMPRESS option is specified in this scenario, the chart uses one horizontal page. If neither the COMPRESS nor PCOMPRESS option is specified, the number of horizontal pages is computed in order to display as much of the chart as possible in a proportional manner.

WEB=*variable*

HTML=*variable*

specifies the character variable in the schedule data set that identifies an HTML page for each activity. The procedure generates an HTML image map using this information for all the schedule bars, milestones, and ID variables corresponding to an activity.

WPREC=*linewidth*

specifies the line width to use for drawing the precedence connections. The default width is 1.

WTNOW=*linewidth*

specifies the line width to use for drawing the timenow line. The default width is 4.

WZONE=*linewidth*

WZLINE=*linewidth*

specifies the line width to use for drawing the horizontal zone lines which demarcate the different zones on the chart. The default linewidth is 1.

ZONE=*variable*

ZONEVAR=*variable*

names the variable in the Schedule data set that is used to separate the Gantt chart into zones. This option enables you to produce a zoned Gantt chart. The GANTT procedure does not sort the Schedule data set and processes the data in the order it appears in the Schedule data set. A change in the value of the zone variable establishes a new zone. By default, the GANTT procedure displays a ZONE variable column before the ID variable columns. You can suppress this column using the NOZONECOL option. The GANTT procedure also draws a horizontal line demarcating zones. By default, the line spans the entire chart in the horizontal direction, both inside and outside the axis area. You can control the span of this line using the ZONESPAN= option. You can also adjust the vertical offset of the line from its default position by using the ZONEOFFSET= option. In addition, you can also control the graphical attributes associated with this line such as color, style, and width using the CZONE=, LZONE=, and WZONE= options, respectively.

ZONEOFF=*d*

ZONEOFFSET=*d*

specifies the offset in cellheights of the zone line from its default position of 0.5 cell height above the top of the first schedule bar for the first activity in the zone. The default value of *d* is 0. The direction of increase is from top to bottom.

ZONESPAN=*name*

ZONELINE=*name*

specifies the span of the horizontal zone line that is drawn at the beginning of each new zone. Valid values for 'name' are LEFT, RIGHT, ALL, and NONE. The value of LEFT draws a line that spans the width of the columns of text that appear on the left hand side of the Gantt chart. The value of RIGHT draws a line that spans the width of the axis area which appears on the right-hand side of the chart. The value of ALL draws a line spanning both the preceding regions while the value of NONE suppresses the line altogether. The default value is ALL.

ID Statement

ID *variables* **;**

The ID statement specifies the variables to be displayed that further identify each activity. If two or more consecutive observations have the same combination of values for all the ID variables, only the first of these observations is plotted unless the DUPOK option is specified in the CHART statement.

By default, if the ID variables do not all fit on one page, they are all omitted and a message explaining the omission is printed to the log. You can override this behavior and display the maximum number of consecutive ID variables that can fit on a page by specifying the MAXIDS option in the CHART statement.

If the time axis of a Gantt chart spans more than one page, the ID variables are displayed only on the first page of each activity. You can display the ID variables on every page by specifying the IDPAGES option in the CHART statement.

Details: GANTT Procedure

Schedule Data Set

Often, the Schedule data set input to PROC GANTT is the output data set (the OUT= data set) produced by PROC CPM, sometimes with additional variables. Typically, this data set contains

- the start and finish times for the early and late schedules (E_START, E_FINISH, L_START, and L_FINISH variables)
- the actual start and finish times (A_START and A_FINISH variables) of activities that have been completed or are in progress for projects that are in progress or completed
- the resource-constrained start and finish times of the activities (S_START and S_FINISH variables) for projects that have been scheduled subject to resource constraints
- the baseline start and finish times (B_START and B_FINISH variables) of activities when monitoring and comparing the progress of a project against a target schedule

When such a data set is input as the Schedule data set to PROC GANTT, the procedure draws a Gantt chart showing five different schedules for each activity: the predicted early/late schedules using E_START, E_FINISH, L_START, and L_FINISH on the first line for the activity, the actual schedule using A_START and A_FINISH on the second line, the resource-constrained schedule using S_START and S_FINISH on the third line, and the baseline schedule using B_START and B_FINISH on the fourth line.

The SEGMT_NO Variable

Normally, each observation of the Schedule data set causes one set of bars to be plotted corresponding to the activity in that observation. If activity splitting has occurred during resource-constrained scheduling, the Schedule data set produced by PROC CPM contains more than one observation for each activity. It also contains a variable named SEGMT_NO. For activities that are not split, this variable has a missing value. For split activities, the number of observations in the Schedule data set is equal to (1 + *the number of disjoint segments that the activity is split into*). The first observation corresponding to such an activity has SEGMT_NO equal to missing, and the S_START and S_FINISH variables are equal to the start and finish times, respectively, of the entire activity. Following this observation, there are as many observations as the number of disjoint segments in the activity. All values for these segments are the same as the first observation for this activity except SEGMT_NO, S_START, S_FINISH, and the duration. SEGMT_NO is the index of the segment, S_START and S_FINISH are the resource-constrained start and finish times for this segment, and duration is the duration of this segment. See the section "Displayed Output" on page 559 for details on how PROC GANTT treats the observations in this case.

NOTE: For a given observation in the Schedule data set, the finish times (E_FINISH, L_FINISH, A_FINISH, S_FINISH, and B_FINISH) denote the last *day* of work when the variables are formatted as SAS *date* values; if they are formatted as SAS *time* or *datetime* values, they denote the last *second* of work. For instance, if an activity has E_START='2JUN04' and E_FINISH='4JUN04', then the earliest start time for the activity is the beginning of June 2, 2004, and the earliest finish time is the end of June 4, 2004. Thus, PROC GANTT assumes that the early, late, actual, resource-constrained, or baseline finish time of an activity is at the end of the time interval specified for the respective variable. The exceptions to this type of default behavior occur when either the DURATION= option or the PADDING= option is in effect. See the section "Specifying the PADDING= Option" on page 529 for further details.

All start and finish times, and additional variables specified in the CHART statement must be numeric and have the same formats. The ID and BY variables can be either numeric or character. Although the data set does not have to be sorted, the output may be more meaningful if the data are in order of increasing early start time. Further, if the data set contains segments of split activities, the data should also be sorted by SEGMT_NO for each activity.

A family of options, available only in graphics mode, enables you to display the precedence relationships between activities on the Gantt chart. The precedence relationships are established by specifying a set of variables in the CHART statement; this can be done in one of two ways. These variables must lie in the Schedule data set and, optionally, in the Precedence data set defined by the PRECDATA= option in the PROC GANTT statement. See the section "Specifying the Logic Options" on page 546 for more details on producing a Logic Gantt chart.

Also available in graphics mode is an automatic text annotation facility that enables you to annotate labels on the Gantt chart independently of the SAS/GRAPH Annotate facility. A useful property of this facility is the ability to link label coordinates and text strings to variables in the Schedule data set. You can create links of two types. An implicit link automatically links an observation in the Label data set to every observation in the Schedule data set. An explicit link uses a variable that must exist on both data sets and be identical in type and length. For more information on the linking variable in the automatic text annotation facility, see the section "Automatic Text Annotation" on page 553.

Missing Values in Input Data Sets

Table 8.2 summarizes the treatment of missing values for variables in the input data sets used by PROC GANTT.

Table 8.2 Treatment of Missing Values in PROC GANTT

Data Set	Variable	Action
CALEDATA	CALID	Default calendar (0 or “DEFAULT”)
	SUN, . . . ,_SAT_	Corresponding shift for default calendar
	D_LENGTH	DAYLENGTH, if available; else, 8:00, if INTERVAL=WORKDAY or DTWRKDAY; 24:00, otherwise
DATA	A_FINISH	Value ignored
	A_START	Value ignored
	ACTIVITY	Input error: logic options are ignored
	B_FINISH	Value ignored
	B_START	Value ignored
	CALID	Default calendar (0 or “DEFAULT”)
	CHART	Value ignored
	DUR	Nonzero
	E_FINISH	Value ignored
	E_START	Value ignored
	HEADNODE	Input error: logic options are ignored
	ID	Missing
	L_FINISH	Value ignored
	L_START	Value ignored
	LAG	FS
	S_FINISH	Value ignored
	S_START	Value ignored
	SEGMT_NO	See the section “Displayed Output” on page 559
	SUCCESSOR	Value ignored
	TAILNODE	Input error: logic options are ignored
	ZONE	Zone value
HOLIDATA	CALID	Holiday applies to all calendars defined
	HOLIDAY	Observation ignored
	HOLIDUR	Ignored, if HOLIFIN is not missing; else, 1.0
	HOLIFIN	Ignored, if HOLIDUR is not missing; else, HOLIDAY + (1 unit of INTERVAL)
LABDATA	_ALABEL	0
	_CLABEL	CTEXT=
	_FLABEL	FONT=
	_HLABEL	1
	_JLABEL	L
	_LABEL	Use _LVAR
	_LVAR	Value ignored
	_PAGEBRK	0
	_RLABEL	0

Table 8.2 (continued)

Data Set	Variable	Action
	_X	Use _XVAR
	_XOFFSET	0
	_XVAR	Value ignored
	_XSYS	DATA
	_Y	Use LABVAR=
	_YOFFSET	0
	_YSYS	DATA
	LABVAR	Value ignored
PRECDATA	ACTIVITY	Input error: logic options are ignored
	LAG	FS
	SUCCESSOR	value ignored
WORKDATA	any numeric variable	00:00, if first observation; 24:00, otherwise

Specifying the PADDING= Option

As explained in the section "Schedule Data Set" on page 526, the finish times in the Schedule data set denote the final time unit of an activity's duration; that is, the activity finishes at the end of the day/second specified as the finish time. A plot of the activity's duration should continue through the end of the final time unit. Thus, if the value of the E_FINISH variable is '4JUN04', the early finish time for the activity is plotted at the end of June 4, 2004 (or the beginning of June 5, 2004).

In other words, the finish times are *padded* by a day (second) if the finish time variables are formatted as SAS date (SAS time or datetime) values. This treatment is consistent with the meaning of the variables as output by PROC CPM. Default values of PADDING corresponding to different format types are shown in Table 8.3.

The PADDING= option is provided to override the default padding explained above. Valid values of this option are NONE, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QTR, YEAR, DTSECOND, DTMINUTE, DTHOUR, DTWEEK, DTMONTH, DTQTR, and DTYEAR. Use the value NONE if you do not want the finish times to be adjusted.

Table 8.3 Default Values of the PADDING= Option Corresponding to Format Type

Format	PADDING
SAS time value	SECOND
SAS date value	DAY
SAS datetime value	DTSECOND
Other	NONE

It is recommended that when plotting zero duration activities, you include a variable in the Schedule data set that has value zero if and only if the activity has zero duration. Defining this variable to the GANTT procedure using the DURATION= (or DUR=) option in the CHART statement ensures that a zero duration activity is represented on the chart by a Milestone. If this is not done, an activity with zero duration is shown on the chart as having a positive duration since finish times are padded to show the end of the last time unit.

Page Format

The GANTT procedure divides the observations (activities) into a number of subgroups of approximately equal numbers. The size of each group is determined by the PAGESIZE system option. Similarly, the time axis is divided into a number of approximately equal divisions depending on the LINESIZE system option.

If the FILL option is specified, however, each page is filled as completely as possible before plotting on a new page. If both axes are split, the pages are ordered with the chart for each group of activities being plotted completely (the time axis occupying several consecutive pages, if needed) before proceeding to the next group.

If a BY statement is used, each BY group is formatted separately.

Two options that control the format of the chart are the MININTERVAL= and SCALE= options. The value for the MININTERVAL= option, denoted by *mininterval*, is the smallest time interval unit to be identified on the chart. The value for the SCALE= option, denoted by *scale*, is the number of columns to be used to denote one unit of *mininterval*. For example, if MININTERVAL=MONTH and SCALE=10, the chart is formatted so that 10 columns denote the period of one month. The first of these 10 columns denotes the start of the month and the last denotes the end, with each column representing approximately three days. Further, the INCREMENT= option can be used to control the labeling. In this example, if INCREMENT=2, then the time axis would have labels for alternate months.

Specifying the MININTERVAL= Option

The value specified for the MININTERVAL= option is the smallest time interval unit to be identified on the chart. If the time values being plotted are SAS *date* values, the valid values for *mininterval* are DAY, WEEK, MONTH, QTR, or YEAR. If the values are SAS *datetime* values, valid values for *mininterval* are DTSECOND, DTMINUTE, DTHOUR, DTDAY, DTWEEK, DTMONTH, DTQTR, or DTYEAR. If they are SAS *time* values, then valid values are SECOND, MINUTE, or HOUR.

NOTE: If the times being plotted are SAS *datetime* values and *mininterval* is either DTSECOND, DTMINUTE, or DTHOUR, the output generated could run into several thousands of pages. Therefore, be careful when choosing a value for *mininterval*.

Table 8.4 shows the default values of *mininterval* corresponding to different formats of the times being plotted on the chart.

Table 8.4 Default Values of the MININTERVAL= Option

Format	MININTERVAL= Value
DATEw.	DAY
DATETIMEw.d	DTDAY
HHMMw.d	HOUR
MONYYw.	MONTH
TIMEw.d	HOUR
YYMMDDw.	MONTH
YYQw.	MONTH

Labeling on the Time Axis

If the variables being plotted in the chart are unformatted numeric values, the time axis is labeled by the corresponding numbers in increments specified by the INCREMENT= option. However, if the variables have *date*, *datetime*, or *time* formats, then the time axis is labeled with two or three lines. Each line is determined by the value of *mininterval*, which in turn is determined by the format of the plotted times (see Table 8.4). Table 8.5 illustrates the format of the label corresponding to different values of *mininterval*.

Table 8.5 Label Format Corresponding to MININTERVAL= Value

MININTERVAL= Value	First Line	Second Line	Third Line
DAY, WEEK, DTWEEK	Month	Day	
MONTH, QTR, YEAR, DTMONTH, DTQTR, DTYEAR	Year	Month	
DTSECOND, DTMINUTE, DTHOUR, DTDAY	Month	Day	Time
SECOND, MINUTE, HOUR	Time		

Multiple Calendars and Holidays

Work pertaining to a given activity is assumed to be done according to a particular *calendar*. A calendar is defined in terms of a *work pattern* for each day and a *workweek structure* for each week. In addition, each calendar may include holidays during the year. See the "Multiple Calendars" section in the PROC CPM chapter for details on how calendars are defined and how all the options work together. In this chapter, a less detailed description is provided. PROC GANTT uses the same structure as PROC CPM for defining calendars, but the options for using them differ in minor ways. The following are the differences in syntax:

- The CALID variable is specified as an option in the CHART statement and is not a separate statement as in PROC CPM.
- The HOLIDAY variable is specified as an option in the CHART statement and is not a separate statement as in PROC CPM.
- The HOLIDUR and HOLIFIN variables are specified as options in the CHART statement and not in a separate HOLIDAY statement.
- The INTERVAL= option is specified in the CHART statement and not in the procedure statement as in PROC CPM.

The WORKDATA (or Workday) data set specifies distinct shift patterns during a day. The CALEDATA (or Calendar) data set specifies a typical workweek for all the calendars in the project; for each day of a typical week, it specifies the shift pattern that is followed. The HOLIDATA (or Holiday) data set specifies a list of holidays and the calendars that they refer to; holidays are defined either by specifying the start of the holiday and its duration in *interval* units, where the INTERVAL= option has been specified as *interval*, or by specifying the start and end of the holiday period. If both the HOLIDUR and the HOLIFIN variables have missing values in a given observation, the holiday is assumed to start at the date and time specified for the HOLIDAY variable and last one unit of *interval*. If a given observation has valid values for both the HOLIDUR and the HOLIFIN variables, only the HOLIFIN variable is used so that the holiday is assumed to start and end as specified by the HOLIDAY and HOLIFIN variables, respectively. The Schedule data set (the DATA= data set) , specifies the calendar that is used by each activity in the project through the CALID variable (or a default variable _CAL_). Each of the three data sets used to define calendars is described in greater detail in the "Multiple Calendars" section in the PROC CPM chapter.

Each new value for the CALID variable in either the Calendar or the Holiday data set defines a new calendar. If a calendar value appears in the Calendar data set and not in the Holiday data set, it is assumed to have the same holidays as the default calendar (the default calendar is defined in the PROC CPM chapter). If a calendar value appears in the Holiday data set and not in the Calendar data set, it is assumed to have the same work pattern structures (for each week and within each day) as the default calendar. In the Schedule data set, valid values for the CALID variable are those that are defined in either the Calendar or the Holiday data set.

All the holiday, workday and workweek information is used by PROC GANTT for display only; in particular, the weekend and shift information is used only if the MARKWKND or MARKBREAK option is in effect. The value of the INTERVAL= option, which has a greater scope in PROC CPM, is used here only to determine the end of holiday periods appropriately. Further, the Workday, Calendar, and Holiday data sets and the processing of holidays and different calendars are supported only when *interval* is DAY, WEEKDAY, WORKDAY, DTSECOND, DTMINUTE, DTHOUR, DTDAY, or DTRKDAY.

Specifying the INTERVAL= Option

The INTERVAL= option is needed only if you want holidays or breaks or both during a week or day to be indicated on the Gantt chart. The value of INTERVAL= is used to compute the start and end of holiday periods to be compatible with the way they were computed and used by PROC CPM. Further, if the MARKWKND or MARKBREAK option is in effect, the INTERVAL= option, in conjunction with the DAYSTART= and DAYLENGTH= options and the Workday, Calendar, and Holiday data sets, helps identify the breaks during a standard week or day as well as the holidays that are to be marked on the chart. Valid values of *interval* are DAY, WEEKDAY, WORKDAY, DTSECOND, DTMINUTE, DTHOUR, DTDAY, and DTWRKDAY. If *interval* is WEEKDAY, WORKDAY, or DTWRKDAY, the MARKWKND option is in effect; otherwise, breaks during a week are indicated only if MARKWKND is specified and breaks within a day are marked only if MARKBREAK is specified.

Full-Screen Version

You can invoke PROC GANTT in full-screen mode by specifying FS (or FULLSCREEN) in the PROC GANTT statement. The full-screen mode offers you a convenient way to browse the Gantt chart for the project. For large projects, where the chart could span several pages, the full-screen mode is especially convenient because you can scroll around the output using commands on the command line, pull-down menus, or function keys. You can scroll vertically to a given job on the task axis by specifying a job number or scroll horizontally to a given point in time along the time axis by specifying a date. You can optionally display the title and the legend.

The specifications for the full-screen version of PROC GANTT and the output format are the same as those for the line-printer version. The following is a list of the few minor differences:

- The FILL option is not relevant in this case because all of the activities are plotted on one logical page.
- The NOLEGEND option is not effective. The screen always displays only the body of the chart along with the ID columns. To see what the symbols mean, you can use the SHOW LEGEND command, which causes the legend to be displayed at the bottom of the chart. To delete the legend, use the DELETE LEGEND command.
- The SUMMARY option is not supported in full-screen mode.
- The SCALE= option works the same way as in the line-printer version, except for its default behavior. The default value is always 1, unlike in the line-printer case where, if the time axis fits on less than one page, the default value is chosen so that the time axis fills as much of the page as possible.

Output Format

The output format is similar to the line-printer version of PROC GANTT. When PROC GANTT is invoked with the FS option, the screen is filled with a display of the Gantt chart. The display consists of column headings at the top and ID values (if an ID statement is used to specify ID variables) at the left. The body of the chart occupies the bottom right portion of the display. The column headings can be scrolled left or right, the ID values can be scrolled up or down, and the body of the chart can scroll along both directions. The display does not include the TITLES or LEGEND.

In addition to using the symbols and join characters as described for the line-printer version of PROC GANTT, the full-screen version also uses different colors to distinguish the types of activities and their associated bars.

You can use the FIND command to locate a particular job (by job number) or a particular time along the time axis. The format of the FIND command is FIND JOB *n* or FIND TIME *t*. All the commands that are specific to PROC GANTT are described as follows.

Local Commands

Table 8.6 lists the commands that can be used in the full-screen version of PROC GANTT.

Table 8.6 Full-Screen Commands and Their Purpose

Scrolling	Controlling Display	Exiting
BACKWARD	SHOW	END
FORWARD	DELETE	CANCEL
LEFT	FIND	
RIGHT		
TOP		
BOTTOM		
VSCROLL		
HSCROLL		

BACKWARD

scrolls towards the top of the Gantt chart by the VSCROLL amount. A specification of BACKWARD MAX scrolls to the top of the chart. You can also specify the vertical scroll amount for the current command as BACKWARD PAGE | HALF | *n*. Note that during vertical scrolling, the column headings are not scrolled.

BOTTOM

scrolls to the bottom of the Gantt chart.

DELETE LEGEND | TITLE

deletes the legend or the title from the screen. A specification of DELETE LEGEND deletes the legend from the current display; DELETE TITLE deletes the current title (titles) from the current display.

END

ends the current invocation of the procedure.

FIND

scrolls to the specified position on the chart. The format of the command is FIND JOB *n* or FIND TIME *t*.

A specification of FIND JOB *n* scrolls backward or forward, as necessary, in order to position the activity with job number *n* on the screen. The specified activity is positioned at the top of the screen, unless this would result in blank space at the bottom of the screen. In this instance, the chart is scrolled down to fit as many jobs as space permits.

A specification of FIND TIME *t* scrolls left or right, as necessary, in order to position the time *t* on the time axis to appear on the screen. The specified time is positioned at the left boundary of the displayed chart area unless this would result in blank space at the right of the screen. In this instance, the chart is scrolled to the right to fit as much of the time axis as space permits.

FORWARD

scrolls towards the bottom of the Gantt chart by the VSCROLL amount. A specification of FORWARD MAX scrolls to the bottom of the chart. You can also specify the vertical scroll amount for the current command as FORWARD PAGE | HALF | *n*. Note that during vertical scrolling, the column headings are not scrolled.

HELP

displays a HELP screen listing all the full-screen commands specific to PROC GANTT.

HOME

moves the cursor to the command line.

HSCROLL

sets the amount of information that scrolls horizontally when you execute the LEFT or RIGHT command. The format is HSCROLL PAGE | HALF | *n*. The specification is assumed to be in number of columns. A specification of HSCROLL PAGE sets the scroll amount to be the number of columns in the part of the screen displaying the plot of the schedules. A specification of HSCROLL HALF is half that amount; HSCROLL *n* sets the horizontal scroll amount to *n* columns. The default setting is PAGE.

KEYS
displays current function key settings.

LEFT
scrolls towards the left boundary of the Gantt chart by the HSCROLL amount. A specification of LEFT MAX scrolls to the left boundary. You can also specify the horizontal scroll amount for the current command as LEFT PAGE | HALF | *n*. Note that during horizontal scrolling, the ID columns are not scrolled.

RIGHT
scrolls towards the right boundary of the Gantt chart by the HSCROLL amount. A specification of RIGHT MAX scrolls to the right boundary. You can also specify the horizontal scroll amount for the current command as RIGHT PAGE | HALF | *n*. Note that during horizontal scrolling, the ID columns are not scrolled.

SHOW LEGEND | TITLE
displays the legend or the title on the screen. A specification of SHOW LEGEND displays the legend in the bottom portion of the current display; SHOW TITLE displays the current title (titles) in the top portion of the current display.

TOP
scrolls to the top of the Gantt chart.

VSCROLL
sets the amount of information that scrolls vertically when you execute the BACKWARD or FORWARD command. The format is VSCROLL PAGE | HALF | *n*. The specification is assumed to be in number of rows. A specification of VSCROLL PAGE sets the scroll amount to be the number of rows in the part of the screen displaying the plot of the schedules. A specification of VSCROLL HALF is half that amount; VSCROLL *n* sets the vertical scroll amount to *n* rows. The default setting is PAGE.

Global Commands

Most of the global commands used in SAS/FSP software are also valid with PROC GANTT. Some of the commands used for printing screens are described below.

SAS/FSP software provides you with a set of printing commands that enable you to take pictures of windows and to route those pictures to a printer or a file. Whether you choose to route these items directly to a printer queue or to a print file, SAS/FSP software provides you with a means of specifying printing instructions. The following is an overview of these related commands and their functions:

FREE
releases all items in the print queue to the printer. This includes pictures taken with the SPRINTcommand as well as items sent to the print queue with the SEND command. All items in the print queue are also automatically sent to the printer when you exit the procedure, send an item that uses a different form, or send an item to a print file. Items are also sent automatically when internal buffers have been filled.

Items sent to a file: If you have routed pictures taken with the SPRINT command to a file rather than to a printer, the file is closed when you execute a FREE command. It is also closed when you send an item that uses a different form, send items to a different print file or to the print queue, or exit the procedure.

NOTE: Any items sent to the same print file after it has been closed will replace the current contents.

PRTFILE *'filename'*

PRTFILE *fileref*

PRTFILE CLEAR

specifies a file to which the procedure sends pictures taken with the SPRINT command instead of sending them to the default printer. You can specify an actual filename or a previously assigned fileref.

Using a filename: To specify a file named *destination-file*, execute

`prtfile` '*destination-file*'

where *destination-file* follows your system's conventions. Note that quotes are required when you specify a filename rather than a fileref.

Using a fileref: You can also specify a previously assigned fileref.

Using the default: Specify PRTFILE CLEAR to prompt the procedure to route information once again to the queue for the default printer.

Identify the current print file: Specify PRTFILE to prompt the procedure to identify the current print file.

SPRINT [NOBORDER][NOCMD]

takes a picture of the current window exactly as you see it, including window contents, border, and command line. By default, the picture is sent to the queue for the default printer.

Border and command line: By default, both the window border and command line are included in the picture you take with the SPRINT command. You can capture a picture of the window contents that excludes either the window border, the command line, or both. Specify the NOBORDER option to exclude the border and the NOCMD option to exclude the command line. Taking a picture of the window contents without the border and command line is a convenient way to print text for a report.

Destination: The destination of the picture captured with the SPRINT command is determined by the PRTFILE command. By default, the picture goes to the default printer. Use the PRTFILE command if you want it sent to a file instead. Each time you execute the SPRINT command, the picture you take is appended to the current print file; it does not write over the current file. See the PRTFILE command for further explanation.

Graphics Version

Formatting the Chart

If necessary, PROC GANTT divides the Gantt chart into several pages. You can force the Gantt chart to fit on one page by specifying the COMPRESS option in the CHART statement. You can achieve a similar result using the PCOMPRESS option, which also maintains the aspect ratio. In addition, you can fit the chart into a prescribed number of horizontal and vertical pages by using the HPAGES= and VPAGES= options in the CHART statement.

The amount of information contained on each page is determined by the values of the graphics options HPOS= and VPOS= specified in a GOPTIONS statement. If any compression of the Gantt chart is performed, the values of HPOS and VPOS are increased, as necessary, to the number of rows and columns respectively, that the entire chart occupies in uncompressed mode. The default height of each row of the Gantt chart is computed as (100/v)% of the screen height where VPOS=v. Thus, the larger the value of VPOS, the narrower the row. You can control the default bar height and default bar offset by using the BARHT= option and the BAROFF= option, respectively. You can further override these at the schedule level. For example, the ABARHT= option affects only the height of the actual schedule bars. The screen is assumed to be divided into h columns where HPOS=h; thus, each column is assumed to be as wide as (100/h)% of the screen width. Hence, the specifications SCALE=10 and MININTERVAL=WEEK imply that a duration of one week is denoted by a bar of length (1000/h)% of the screen width.

The height of the text characters is controlled by both the HEIGHT= option in the CHART statement and the HTEXT= option specified in a GOPTIONS statement. The text height is set equal to the product of the HEIGHT= and HTEXT= values. (If neither the HEIGHT= option nor the HTEXT= option has been specified, the text height is given by the font size attribute of the GraphDataText element of the current ODS style template. See the section "ODS Style Templates" on page 562 for more information about ODS styles.) The units in which the text height is measured are those of the HTEXT= option. By default, the value of HEIGHT= is 1, which sets the text height to be equal to the HTEXT= value. The default value of HTEXT= is 1 unit, where a unit is defined by the GUNIT= option in a GOPTIONS statement. Thus, in the absence of the HEIGHT=, HTEXT=, and GUNIT= options, and with no font size provided by the current ODS style template, the text height is the same as the bar height, namely one cell height. Increasing the value of HEIGHT= is useful when you use the COMPRESS option, particularly when you have a very large chart. Since the chart is scaled as appropriate to fit on one page, the text can be very hard to discern, or even illegible, and would benefit from enlargement. Relative positioning of the font baseline for activity text is controlled by the HTOFF= option in the CHART statement. By default, the font baseline for an activity is at the bottom of the first bar corresponding to the activity.

The color of the text characters is specified by using the CTEXT= option in the CHART statement. The default color depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 562 for more information. You can override the text colors for selected columns of activity text at the activity level by using a PATTERN variable in the Schedule data set and specifying the CTEXTCOLS= option in the CHART statement.

The font used for the text characters is specified with the FONT= option in the CHART statement. The default font depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 562 for more information.

Global PATTERN statements are used to specify the fill pattern for the different types of bars drawn on the Gantt chart. Each fill pattern can be associated with a color. Patterns can be used to reflect the status of an activity (normal, critical, supercritical) in the predicted early/late schedule, to indicate the different schedule types (actual, resource-constrained, baseline), and to represent weekends, holidays and breaks on the Gantt chart. See the section "Using PATTERN Statements" on page 538 for details. In addition, you can override these fill patterns for selected schedules at an activity level by using a PATTERN variable in the Schedule data set and specifying the PATLEVEL= option in the CHART statement.

You can use global SYMBOL statements to define the symbols that represent CHART variables in the Gantt chart. The SYMBOL statement enables you to select symbols from different fonts and modify their appearance to suit your requirements. You can specify a color and a height for the symbol in addition to a variety of other options. See the section "Using SYMBOL Statements" on page 542 for details.

Annotate Processing

The Annotate facility enables you to enhance graphics output produced by PROC GANTT. However, if the only items being annotated are symbols and text strings, it is recommended that you use the Automatic Text Annotation facility that is built into the Gantt procedure instead. This facility was developed specifically for labeling Gantt charts; it has some very useful features and requires a minimum of effort.

To use the SAS/GRAPH Annotate facility, you must create an Annotate data set that contains a set of graphics commands that can be superimposed on the Gantt chart. This data set has a specific format and must contain key variables. Each observation in the Annotate data set represents a command to draw a graphics element or perform an action. The values of the variables in the observation determine what is done and how it is done. The observations in an Annotate data set can be created by explicitly assigning values to the Annotate variables through a DATA step or SAS/FSP procedure or by implicitly assigning values with Annotate macros within a SAS DATA step. The process of creating Annotate observations is greatly simplified through the use of Annotate macros.

Coordinates specify where graphic elements are to be positioned. A coordinate system, in turn, determines how coordinates are interpreted. There are several different coordinate systems that are used by the Annotate facility. Typically, one of three major drawing areas can be associated with any coordinate system: data area, procedure output area, and graphics output area. This chapter explains the coordinate system that is based on the data area of PROC GANTT.

When annotating a graph produced by any of the graphics procedures, you may find it helpful to use data coordinates that refer to the data values corresponding to the graph that is being annotated. For example, if you want to label a particular activity of a Gantt chart with additional text, you can position the text accurately if you use data coordinates instead of screen coordinates. With respect to PROC GANTT, the Annotate facility uses the time axis and the activity axis of the Gantt chart as the basis for the data coordinate system. To use this feature, create a Annotate data set based on the Schedule data set that is input to the procedure, utilizing Annotate macros whenever possible to simplify the process.

NOTE: The data coordinate system enables you to annotate the graph even if it spans multiple pages. However, each annotation must be entirely contained within a given page. For example, you cannot annotate a line on the Gantt chart that runs from one page of the chart to another.

In addition to a coordinate system based on the data, you can select a coordinate system based on either the procedure output area or the Graphics output area. You would typically need to use one of these systems, for example, if you want to annotate text outside the chart area.

Using PATTERN Statements

PROC GANTT uses those patterns that are available with the GCHART procedure. PROC GANTT uses a maximum of nine different patterns to denote various phases in an activity's duration and the various types of schedules that are plotted. Patterns are specified in PATTERN statements that can be used anywhere in your SAS program. Table 8.7 lists the function of each of the first nine PATTERN statements that are used by PROC GANTT.

Any PATTERN statements that you specify are used. If more are needed, default PATTERN statements are used.

You can override any of these patterns at the activity level by using a PATTERN variable in the schedule data set. A PATTERN variable is identified by specifying the PATTERN= option in the CHART statement or by the presence of the default _PATTERN variable.

Table 8.7 PATTERN Statements used by PROC GANTT

PATTERN	Used to Denote
1	Duration of a noncritical activity
2	Slack time for a noncritical activity
3	Duration of a critical activity
4	Slack time for a supercritical activity
5	Duration of a supercritical activity
6	Actual duration of an activity
7	Break due to a holiday
8	Resource-constrained duration of an activity
9	Baseline duration of an activity

Refer to the SAS/GRAPH documentation for a detailed description of PATTERN statements. Most of the relevant information is reproduced here for the sake of completeness.

PATTERN Statement Syntax

The general form of a PATTERN statement is

PATTERN*n options;*

where

- *n* is a number ranging from 1 to 255. If you do not specify a number after the keyword PATTERN, PATTERN1 is assumed.
- *options* enables you to specify the colors and patterns used to fill the bars in your output.

PATTERN statements are additive; if you specify a C= or V= option in a PATTERN statement and then omit that option in a later PATTERN statement ending in the same number, the option remains in effect. To turn off options specified in a previous PATTERN*n* statement, either specify all options in a new PATTERN*n* statement, or use the keyword PATTERN*n* followed by a semicolon. For example, the following statement turns off any C= or V= option specified in previous PATTERN3 statements:

```
pattern3;
```

You can reset options in PATTERN statements to their default values by specifying a null value. A comma can be used (but is not required) to separate a null parameter from the next option.

For example, both of the following statements cause the C= option to assume its default value (the value of the CPATTERN= option or the first color in the COLORS= list):

```
pattern c=, v=solid;
```

or

```
pattern c= v=solid;
```

In the following statement, both options are reset to their default values:

```
pattern2 c= v=;
```

You can also turn off options by specifying the RESET= option in a GOPTIONS statement.

General options

You can specify the following options in a PATTERN statement.

COLOR= *color*

C= *color*

specifies the color to use for a bar or other area to be filled. If you do not specify the C= option in a PATTERN statement, the procedure uses the value you specified for the CPATTERN= option in a GOPTIONS statement. If you omitted the CPATTERN= option, the procedure uses the pattern specified by the V= option (see below) with each color in the COLORS= list before it uses the next PATTERN statement.

REPEAT= *n*

R= *n*

specifies the number of times the PATTERN statement is to be reused. For example, the following statement represents one pattern to be used by SAS/GRAPH software:

```
pattern1 v=x3 c=red;
```

You can use the REPEAT= option in the statement to repeat the pattern before going to the next pattern. For example, if you specify the following statements, PATTERN1 is repeated ten times before PATTERN2 is used:

```
pattern1 v=x3 c=red  r=10;
pattern2 v=s  c=blue r=10;
```

Remember that if you omit the COLOR= option in the PATTERN statement and you do not specify the CPATTERN= option, SAS/GRAPH software repeats the pattern for each color in the current COLORS= list. If you specify the R= option in a PATTERN statement from which the C= option is omitted, the statement cycles through the COLORS= list the number of times given by the value of the R= option.

For example, if the current device has seven colors, then the following statement results in 70 patterns because each group of seven patterns generated by cycling through the COLORS= list is repeated ten times:

```
pattern v=x3 r=10;
```

VALUE= *value*

V= *value*

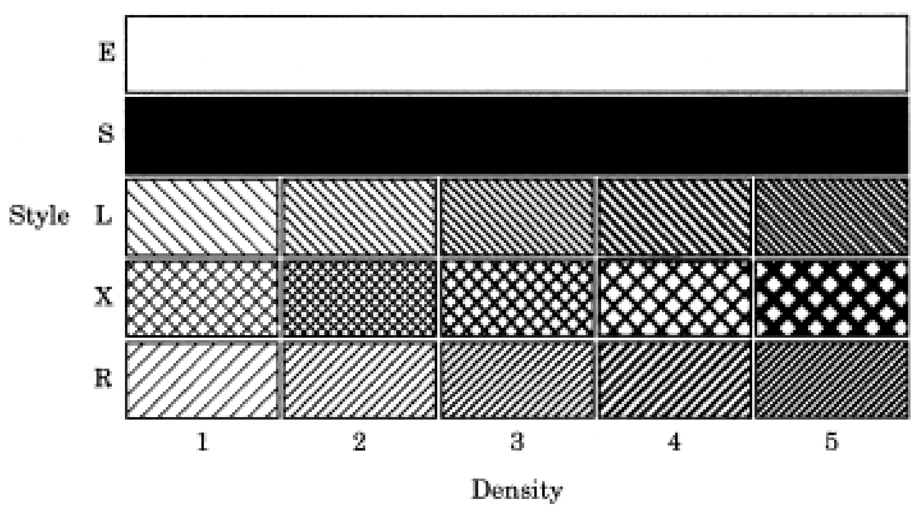
specifies the pattern to use for a bar or other area to be filled. The valid values you can use depend on what procedure you are using and the type of graph you are producing. In PROC GANTT, which produces bars, you must use one of the pattern values shown in Figure 8.6.

In a PATTERN statement, if you specify a value for the V= option but not for the C= option, the procedure uses the value you specified for the CPATTERN= option in a GOPTIONS statement. If you omitted the CPATTERN= option, the procedure uses the pattern specified for the V= option with each color in the COLORS= list before it uses the next PATTERN statement. Thus, if you specify the following statements, the PATTERN1 statement is used for the first type of bar, namely, for the duration of a noncritical activity:

```
pattern1 c=red   v=x3;
pattern2         v=s;
pattern3 c=blue  v=l3;
pattern4 c=green v=r4;

proc gantt data=sched;
```

The PATTERN2 statement is used for the second type of bar, namely, for the slack time of a noncritical activity. Because a C= value is not specified in the PATTERN2 statement, SAS/GRAPH software uses the PATTERN2 statement and cycles through the colors in the COLORS= list for the device to obtain as many patterns as there are colors in the list. If needed, the PATTERN3 and PATTERN4 values are then used for any remaining types of bars.

Figure 8.6 Pattern Selection Guide



Using SYMBOL Statements

You can specify a SYMBOL statement anywhere in your SAS program. SYMBOL statements give PROC GANTT information about the characters to be used for plotting the CHART variables.

See also the section "Special Fonts for Project Management and Decision Analysis" on page 544 for a description of some typically used Gantt chart symbols that can be specified using a SYMBOL statement.

Refer to the SAS/GRAPH documentation for a detailed description of SYMBOL statements. Most of the relevant information is reproduced here for the sake of completeness.

SYMBOL Statement Syntax

The general form of a SYMBOL statement is

SYMBOL*n options;*

where

- *n* is a number ranging from 1 to 255. Each SYMBOL statement remains in effect until you specify another SYMBOL statement ending in the same number. If you do not specify a number following the keyword SYMBOL, SYMBOL1 is assumed.
- *options* enables you to specify the plot characters and color.

SYMBOL statements are additive; that is, if you specify a given option in a SYMBOL statement and then omit that option in a later SYMBOL statement ending in the same number, the option remains in effect. To turn off all options specified in previous SYMBOL statements, you can specify all options in a new SYMBOL*n* statement, use the keyword SYMBOL*n* followed by a semicolon, or specify a null value. A comma can be used (but is not required) to separate a null parameter from the next option.

For example, both of the following statements cause the C= option to assume its default value (the value of the CSYMBOL= option or the first color in the COLORS= list):

```
symbol1 c=, v=plus;
```

and

```
symbol1 c= v=plus;
```

In the following statement, both options are reset to their default values:

```
symbol4 c= v=;
```

You can also turn off options by specifying the RESET= option in a GOPTIONS statement.

General options

You can specify the following options in the SYMBOL statement.

COLOR=*color*

C=*color*

specifies the color to use for the corresponding plot specification. If you do not specify the C= option in a SYMBOL statement, the procedure uses the value you specified for the CSYMBOL= option in a GOPTIONS statement. If you omit the CSYMBOL= option, the procedure uses the value specified by the V= option with each color in the COLORS= list before it uses the next SYMBOL statement.

FONT=*font*

F=*font*

specifies the font from which the symbol corresponding to the value specified with the V= option is to be drawn. If you do not specify a font, the V= option specifies the symbol from the special symbol table shown in Figure 8.7.

H=*height*

specifies the height of the symbol that is to be drawn.

For example, this SYMBOL statement

```
symbol1 c=green v=K f=special h=2;
```

indicates that the symbol at each data point is the letter K from the SPECIAL font (a filled square), drawn in green, the height being twice the bar height.

REPEAT=*n*

R=*n*

specifies the number of times the SYMBOL statement is to be reused.

V=*special-symbol*

V=*'string'*

identifies the symbols from the font specified by the FONT= option in the SYMBOL statement for the corresponding plot specifications. If the FONT= option is not specified, the plot symbol is the symbol corresponding to the value of V= in the special symbol table shown in Figure 8.7. Also permitted without a FONT= specification are the letters A through W and the numbers 0 through 9. If the font is a symbol font, such as MARKER, the string specified with the V= option is the character code for the symbol. If the font is a text font, such as SWISS, the string specified with the V= option is displayed as the plot symbol. By default, the value of V= is PLUS, which produces the plus symbol (+) from the special symbol table.

Note that if you use the special symbol comma (`,`) with the V= option, you must enclose the comma in quotes as illustrated in the following statement:

```
symbol1 v=',';
```

Figure 8.7 Special Symbol Table

VALUE=		Plot Symbol	VALUE=		Plot Symbol
PLUS		+	%	(percent)	♧
X		×	&	(ampersand)	☘
STAR		∗	'	(single quote)	⚜
SQUARE		□	=	(equals)	☆
DIAMOND		◇	-	(hyphen)	☉
TRIANGLE		△	@	(at)	☿
HASH		#	*	(asterisk)	♀
Y		Y	+	(plus)	⊕
Z		Z	>	(greater than)	♂
PAW		∷	.	(period)	♃
POINT		·	<	(less than)	♄
DOT		●	,	(comma)	♅
CIRCLE		○	/	(slash)	♆
_	(underscore)		?	(question mark)	♇
"	(double quote)	♤	(	(left parenthesis)	☾
#	(pound sign)	♡	)	(right parenthesis)	☄
$	(dollar sign)	◇	:	(colon)	✳

Note: The words or special characters in the VALUE= column are entered exactly as shown.

Special Fonts for Project Management and Decision Analysis

Two special marker fonts, ORFONT and ORFONTE, are available in versions 6.08 and later. These two fonts are meant to be used with SAS/OR software and provide a variety of symbols that are typically used in Project Management and Decision Analysis. The fonts ORFONT and ORFONTE are shown in Figure 8.8 and Figure 8.9, respectively. The fonts behave like any SAS/GRAPH font providing you with the capability to control attributes such as color and height.

Figure 8.8 ORFONT - A Filled Font

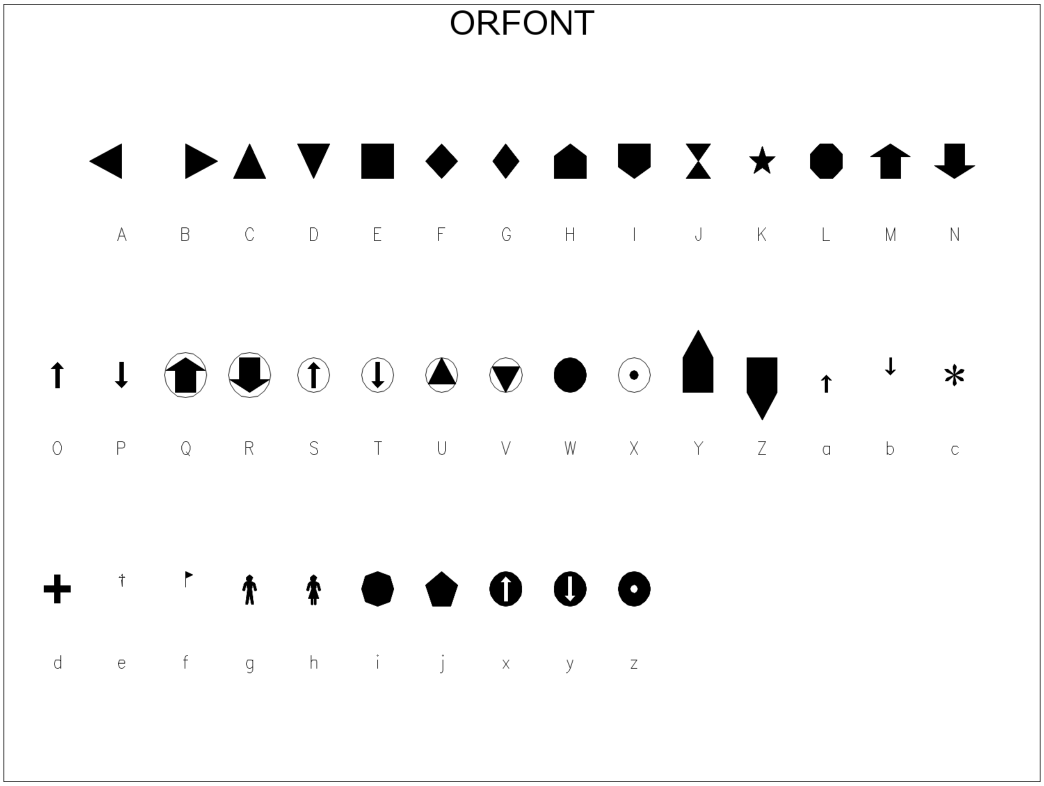
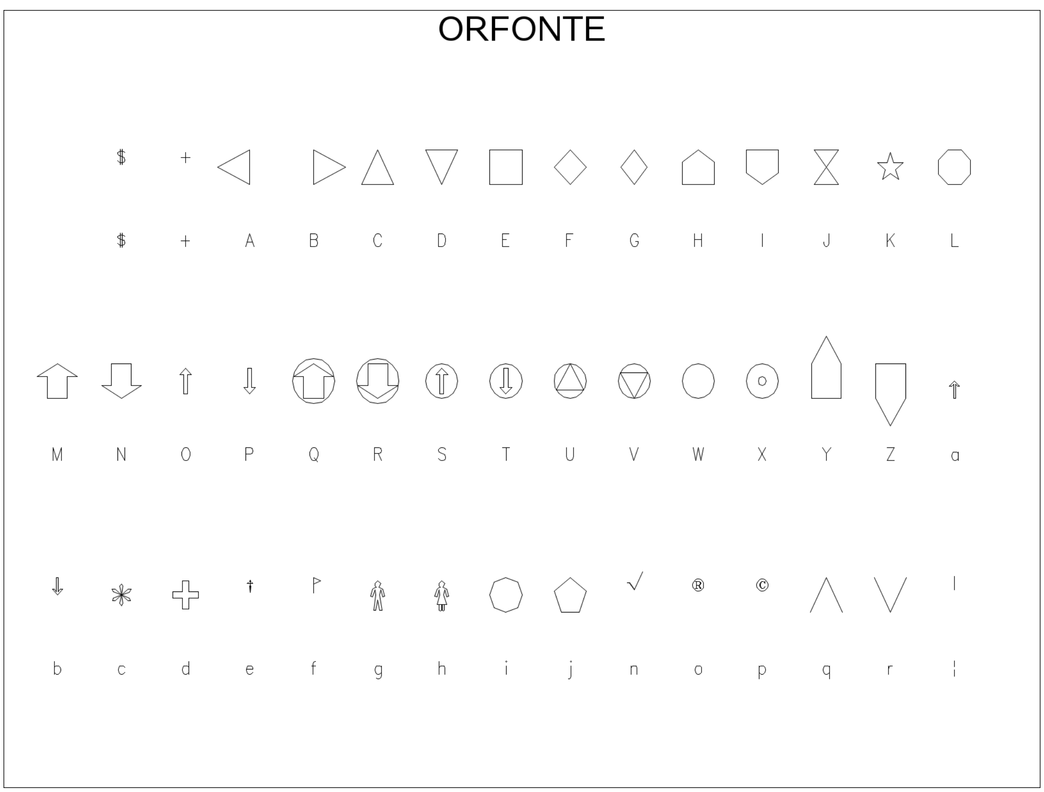


Figure 8.9 ORFONTE - An Empty Font



For example, to use a filled yellow "doghouse" symbol to represent milestones on the Gantt chart, specify the options

```
VMILE="H"  FMILE=ORFONT  CMILE=YELLOW
```

in the CHART statement.

If you wish to represent a CHART variable with an empty blue "circled arrow," then specify the following options in the corresponding SYMBOL statement.

```
V="Q"  F=ORFONTE  C=BLUE;
```

Specifying the Logic Options

The Logic options are a family of options used with the GANTT procedure that enable you to view the precedence relationships between activities on the Gantt chart. The Logic options constitute a high-resolution graphics feature and, as such, are only valid with specification of the GRAPHICS option in the PROC GANTT statement. The Logic options can accommodate nonstandard precedence relationships. The Logic options enable you to control the color, line style, and width of the connecting arcs as well as their layout and positioning on the Gantt chart. You can specify the precedence information required to draw the connections in one of two formats and store it in a data set different from the Schedule data set. You can also use the Schedule data set produced by PROC CPM to provide the precedence information. When using the Schedule data set from PROC CPM, you can ensure that all the relevant precedence information exists in the data set by either specifying the XFERVARS option in the PROC CPM statement or by using an ID statement.

The Logic options are not valid with the specification of either a BY statement or the COMBINE option in the CHART statement.

In order to invoke the logic options, you need to, minimally, specify a set of variables that defines the precedence relationships between tasks. This can be done using one of two formats for defining project networks, the AOA specification or the AON specification.

Activity-on-Arc (AOA) Specification

In the AOA specification, each activity of the project is represented by an arc. The node at the tail of the arc represents the start of the activity, and the node at the head of the arc represents the finish of the activity. The relationship between an activity and its successor is represented by setting the tail node of the successor arc to be the head node of the activity arc. One of the disadvantages of using the AOA method is that it cannot accommodate nonstandard lag types; all lag types are of the Finish-to-Start (FS) type.

The variables required by PROC GANTT to establish a valid AOA specification are defined using the HEADNODE= and TAILNODE= options in the CHART statement.

Activity-on-Node (AON) Specification

In the AON specification, each activity is represented by a node. All arcs originating from an activity terminate at its successors. Consequently, all arcs terminating at an activity originate from its predecessors.

The variables required by PROC GANTT to establish a valid AON specification are defined by the ACTIVITY= and SUCCESSOR= options in the CHART statement.

Optionally, nonstandard precedence relationships can be specified using the LAG= option in the CHART statement to define a variable that defines the lag type of a relationship.

Precedence Data Set

When using the AON specification, you can specify the precedence information using a data set different from the Schedule data set. This is particularly useful when producing several Gantt charts for the same project with different schedule information as would typically be the case when monitoring a project in progress. It eliminates the requirement that the precedence information exist in each Schedule data set and enables for more compact data. This separate data set is specified by the PRECDATA= option in the PROC GANTT statement statement and is referred to as the *Precedence data set*.

In order to graphically represent the precedence relationships derived from the Precedence data set on the Gantt chart, you must link the Precedence data set with the Schedule data set by means of a common variable. This common variable is selected as the ACTIVITY variable by virtue of the fact that it always exists in the Precedence data set. Thus, when using the Precedence data set, you need to ensure that the ACTIVITY variable exists in the Schedule data set, too.

In the event that both a valid AOA and a valid AON specification exist, PROC GANTT uses the AON specification by default. To override the default, use the AOA option in the CHART statement.

Drawing the Precedence Connections

The relationship between an activity and its successor is represented on the Gantt chart by a series of horizontal and vertical line segments that connect their schedule bars corresponding to a specified type (early/late, actual, and so forth). For a given connection, the intersection of a horizontal segment with a vertical segment is called a *turning point* of the connection. The type of the schedule bar used for the connection, also called the *logic bar*, is determined by the LEVEL= option in the CHART statement.

Every connection is comprised of either three or five segments and is termed a 3-segment or a 5-segment connection, respectively. The segments are routed in the following sequence:

a) a horizontal segment that originates from the appropriate end of the logic bar corresponding to the activity. The length of this segment is controlled by the MINOFFGV= and MININTGV= options in the CHART statement.

b) a vertical segment traveling from activity to the successor

c) a horizontal segment traveling towards the appropriate end of the successor's logic bar. The length of this segment is determined by the MINOFFLV= and MAXDISLV= options in the CHART statement.

d) a vertical and horizontal segment into the logic bar of the successor

Every connection begins with a horizontal line segment originating from the activity's logic bar and ends with a a horizontal line segment terminating at the successor's logic bar. If the lag type of the relationship is SS or SF, the initial horizontal segment originates from the left end of the activity's logic bar, otherwise it originates from the right end of the logic bar. If the lag type of the relationship is SS or FS, the final horizontal segment terminates at the left end of the successor's logic bar, otherwise it terminates at the right end of the logic bar.

NOTE: The ends of the bars must be consistent with the lag type of the connection if it is to be drawn; that is, the left end of the activity's logic bar must represent a start time if an SS or SF lag type connection is to be drawn, and the right end of the activity's logic bar must represent a finish time if an FS or FF lag type connection is to be drawn.

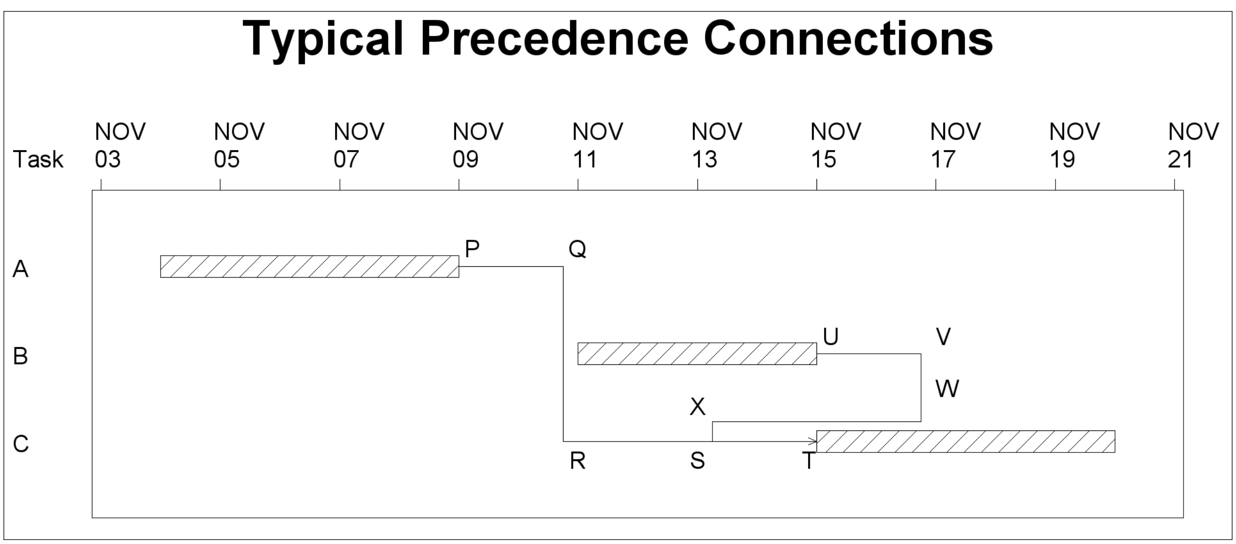
Violation of these conditions is unlikely when using the Schedule data set generated by PROC CPM. An example violating these conditions is a Schedule data set containing incorrect or invalid data. The following example illustrates two observations that are in violation of these conditions. The first observation is invalid data (E_START greater than E_FINISH) while the second observation is incomplete (missing E_START and L_FINISH times).

E_START	E_FINISH	L_START	L_FINISH
03MAR04	01MAR04	.	.
.	05MAR04	07MAR04	.

The following figure illustrates two typical precedence connections between an activity and its successor.

ACTIVITY	SUCCESSOR	LAG
A	C	FS
B	C	FS

Figure 8.10 Typical Precedence Connections



The connection from activity **A** to activity **C** is comprised of three segments PQ, QR, and RT whereas the connection from activity **B** to activity **C** is made up of five segments UV, VW, WX, XS, and ST; the two additional segments correspond to the optional segments mentioned in item **d)** above. Points Q, R, V, W, X, and S are turning points.

Formatting the Axis

If neither MINDATE= nor MAXDATE= have been specified, the time axis of the Gantt chart is extended by a small amount in the appropriate direction or directions in an attempt to capture all of the relevant precedence connections on the chart. While this will succeed for the majority of Gantt charts, it is by no means guaranteed. If connection lines still tend to run off the chart, you can perform one or both of the following tasks.

- Use the MINDATE= or MAXDATE= options (or both) in the CHART statement to increase the chart range as necessary.
- Decrease the values of the MINOFFGV=, MININTGV=, MAXDISLV=, and MINOFFLV= options to reduce the horizontal range spanned by the vertical segments so that they will lie within the range of the time axis.

On the other hand, if the automatic extension supplied by PROC GANTT is excessive, you can suppress it by specifying the NOEXTRANGE option in the CHART statement.

The following section, "Controlling the Layout," addresses the CHART statement options MINOFFGV=, MININTGV=, MINOFFLV=, and MAXDISLV= which control placement of the vertical segments that make up a connection. For most Gantt charts, default values of these options will suffice since their usage is typically reserved for "fine tuning" chart appearance. This section can be skipped unless you want to control the layout of the connection. The description of the layout methodology and concepts is also useful to help you understand the routing of the connections in a complex network with several connections of different types.

Controlling the Layout

The concepts of global and local verticals are first introduced in order to describe the function of the segment placement controls.

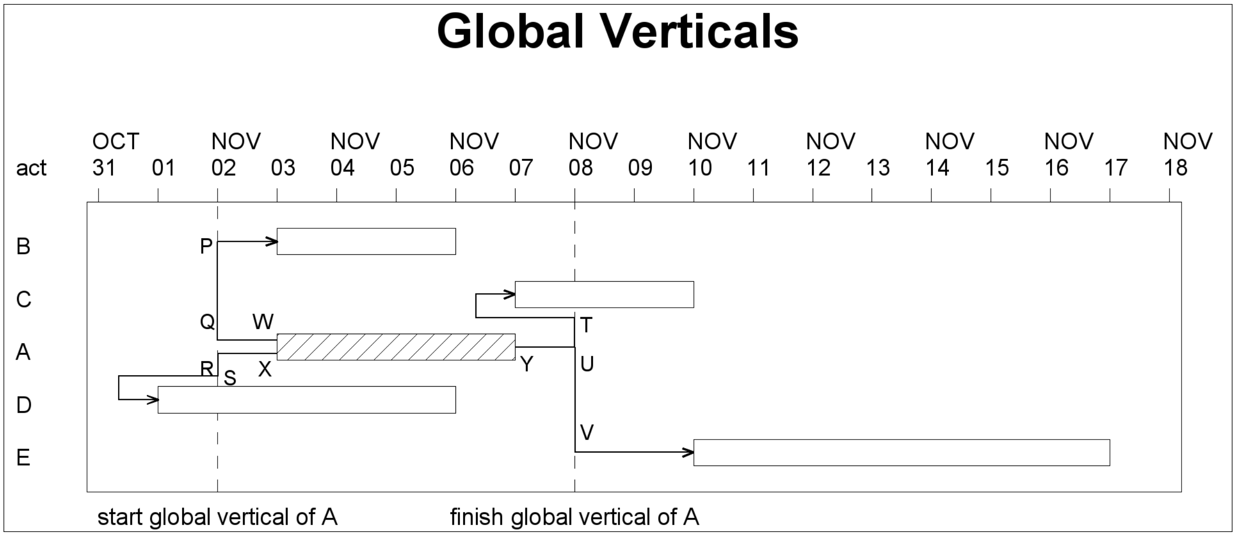
Global Verticals

In the interest of minimizing clutter on the chart, each activity is assigned a maximum of two vertical tracks for *placement* of the vertical segment described in item **b)** above. One vertical track is maintained for SS and SF lag type connections and is referred to as the *start global vertical* of the activity, while the other vertical track is maintained for FS and FF lag type connections and is referred to as the *finish global vertical* of the activity. The term *global vertical* refers to either start global vertical or finish global vertical.

NOTE: The use of the term "global" is attributed to the fact that in any connection from an activity to its successor, the global vertical of the activity corresponds to the *only segment* that travels from activity to successor.

Figure 8.11 illustrates the two global verticals of activity **A**.

Figure 8.11 Global Verticals



Activity **A** has four successors: activities **B**, **C**, **D**, and **E**. The lag type of the relationship between **A** and **B** is nonstandard, namely 'Start-to-Start', as is that between **A** and **D**. The other two lag types are standard. The start and finish global verticals of activity **A** are represented by the two dotted lines. The vertical segments of the SS lag type connections from **A** to **B** and from **A** to **D** that are placed along the start global vertical of **A** are labeled PQ and RS, respectively. The vertical segments of the FS lag type connections from **A** to **C** and from **A** to **E** that are placed along the finish global vertical of **A** are labeled TU and UV, respectively.

For a given connection from activity to successor, the vertical segment that is placed on the activity global vertical is connected to the appropriate end of the logic bar by the horizontal segment described in item **a)** above. The minimum length of this horizontal segment is specified with the MINOFFGV= option in the CHART statement. Further, the length of this segment is affected by the MININTGV= option in the CHART statement, which is the minimum interdistance of any two global verticals. In Figure 8.11, the horizontal segments QW and RX connect the vertical segments PQ and RS, respectively, to the logic bar and the horizontal segment YU connects both vertical segments TU and UV to the logic bar.

Local Verticals

Each activity has seven horizontal tracks associated with it, strategically positioned on either end of the logic bar, above the first bar of the activity, and below the last bar of the activity. These tracks are used for the *placement* of the horizontal segments described in items **c)** and **d)**, respectively.

Figure 8.12 illustrates the positions of the horizontal tracks for an activity in a Gantt chart with four schedule bars. Three of the horizontal tracks, namely **track 1**, **track 4**, and **track 7**, service the start of the logic bar and are connected to one another by a vertical track referred to as the *Start Local Vertical*. Similarly, the horizontal tracks **track 2**, **track 3**, **track 5**, and **track 6** service the finish of the bar and are interconnected by a vertical track referred to as the *Finish Local Vertical*. The local verticals are used for *placement* of the vertical segment described in item **d)** above.

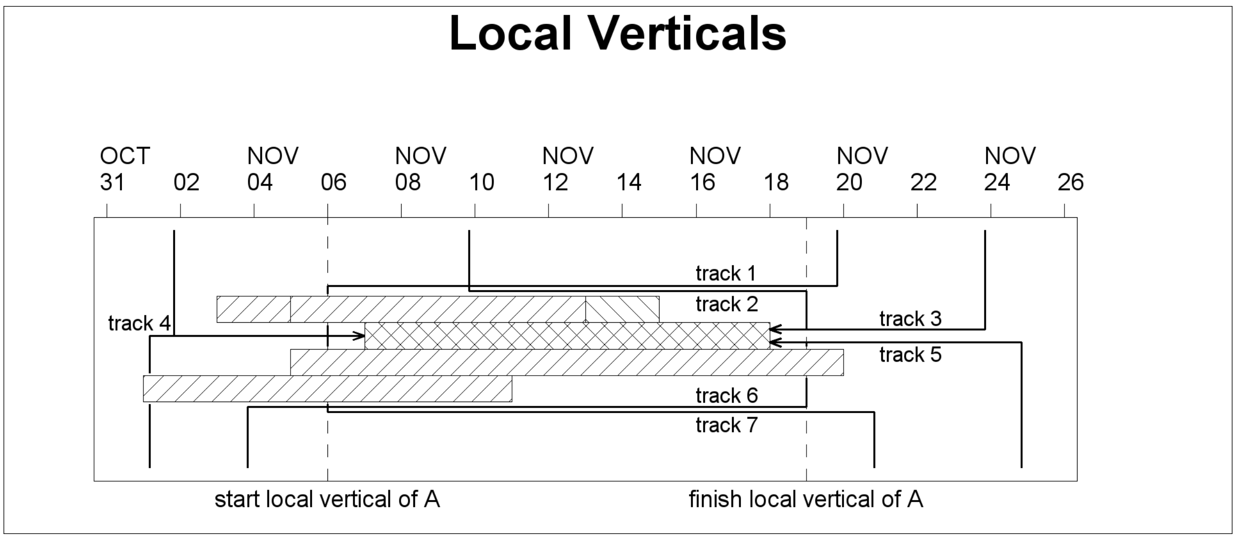
NOTE: The use of the term "local" is attributed to the fact that the local vertical is used to connect horizontal tracks associated with the *same* activity.

Notice that **`track 1`** and **`track 7`** terminate upon their intersection with the start local vertical and that **`track 2`** and **`track 6`** terminate upon their intersection with the finish local vertical.

The minimum distance of a local vertical from its respective bar end is specified with the MINOFFLV= option in the CHART statement. The maximum displacement of the local vertical from this point is specified using the MAXDISLV= option in the CHART statement. The MAXDISLV= option is used to offset the local vertical in order to prevent overlap with any global verticals.

Arrowheads are drawn by default on the horizontal tracks corresponding to the logic bar, namely **`track 3`**, **`track 4`**, and **`track 5`**, upon entering the bar and on continuing pages. The NOARROWHEAD option is used to suppress the display of arrowheads.

Figure 8.12 Local Verticals



Routing the Connection

The routing of the precedence connection from an activity to its successor is dependent on two factors, namely

- the horizontal displacement of the appropriate global vertical of the activity relative to the appropriate local vertical of the successor
- the vertical position on the task axis of the activity relative to the successor

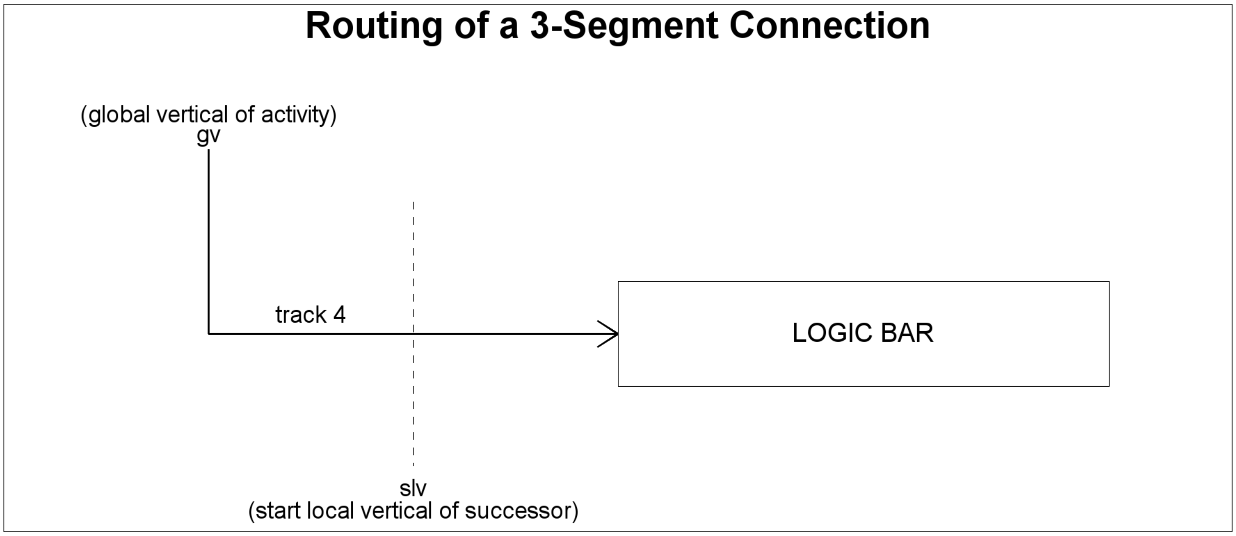
The routing of a SS or FS type precedence connection from activity to successor is described below. A similar discussion holds for the routing of a SF or FF type precedence connection.

Suppose the activity lies above the successor. Let the start local vertical of the successor be denoted by *slv*, and let the appropriate global vertical of the activity be denoted by *gv*.

Case 1:

If *gv* lies to the left of *slv*, then the connection is routed vertically down along *gv* onto **`track 4`** of the successor, on which it is routed horizontally to enter the bar. The resulting 3-segment connection is shown in Figure 8.13.

Figure 8.13 3-Segment Connection

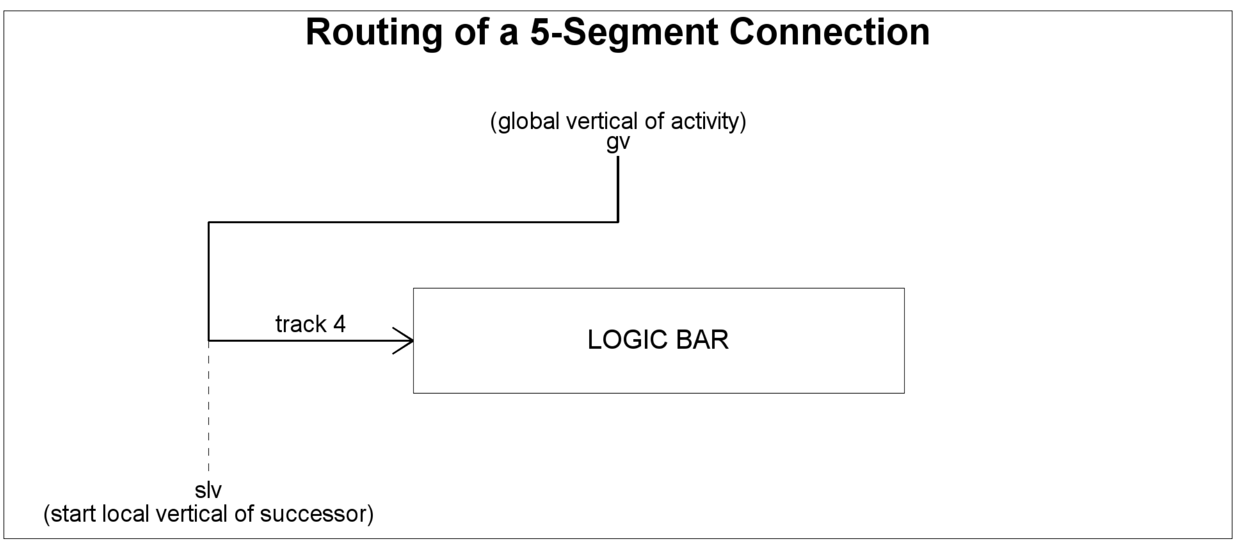


An example of this type of routing is illustrated by the connection between activities **`A`** and **`C`** in Figure 8.10.

Case 2:

If *gv* lies to the right of *slv*, then the connection is routed vertically down along *gv* onto **`track 1`** of the successor, horizontally to the left to meet *slv*, vertically down along *slv* onto **`track 4`** of the successor and horizontally to the right to enter the bar. The resulting 5-segment connection is shown in Figure 8.14.

Figure 8.14 5-Segment Connection



This type of routing is illustrated by the connection between activities **`B`** and **`C`** in Figure 8.10.

An identical description applies when the activity lies below the successor, with the only difference being that **`track 7`** is used in place of **`track 1`** (see Figure 8.12).

Automatic Text Annotation

The automatic text annotation feature is designed specifically for labeling Gantt charts independently of the SAS/GRAPH Annotate facility. This facility enables you to display label strings with a minimum of effort and data entry while providing the capability for more complex chart labeling situations. Some of the properties that characterize this feature are

- the ability to tag labels. This enables you to define 1-1, 1-many, many-1, and many-many relationships.
- the ability to link label coordinates and label strings to variables in the Schedule data set. This enables the Label data set to remain unchanged even if the Schedule data set changes, such as when monitoring a project.
- the ability to automatically format or convert numeric variable values that have been specified for label text strings
- the ability to automatically split strings embedded with blanks to make the pieces as equal in length as possible, with the provision to override this behavior by specifying a split character
- the ability to mix data and percentage coordinates
- the ability to clip labels running off the frame of the Gantt chart

All relevant information is contained in a SAS data set specified using the LABDATA= data set option in the PROC GANTT statement. This data set is also referred to as the Label data set in the context of this documentation. The Label data set is required to contain certain variables in order to determine the label string and the positional information related to the string. At the very least, it requires three variables, one to determine the string to be displayed, one to determine the horizontal position, and one to determine the vertical position. The procedure terminates if it cannot find the required variables.

Table 8.8 Special Symbol Table

Determining the ...	**requires the following variables**
Label text string	_LVAR and/or _LABEL
Horizontal placement position	_XVAR and/or _X
Vertical placement position	LABVAR= and/or _Y

The LABVAR variable refers to the variable specified with the LABVAR= option in the CHART statement. It is the LABVAR variable that links the Schedule and Label data sets together. As far as possible, the procedure attempts to use the _X, _Y, and _LABEL variables in the Label data set. However, a link established using the LABVAR variable makes the Schedule data set a secondary source of information for determining positional and text string information for linked observations. The exact meaning of the preceding variables is explained later in this chapter.

Note that, other than the preceding requirements, there are no further restrictions on the Label data set. In fact, the Schedule data set can also be specified as the Label data set as long as the required variables are present. There are several optional variables in the Label data set. These variables enable you to specify offsets in both horizontal and vertical directions from the given coordinate position; adjust graphical attributes such as baseline angles, character rotations, colors, fonts, and heights; control justification of strings; control placement behavior at pagebreaks; and specify coordinate reference systems for the horizontal and vertical values.

Label Data Set

You specify the Label data set using the LABDATA= option in the PROC GANTT statement. This initiates the labeling of the Gantt chart. The Label data set contains the information that provides the means of determining the label strings and their placement positions. As far as possible, the procedure attempts to use the _X, _Y, and _LABEL variables in the Label data set to extract the horizontal position, the vertical position, and the text string, respectively. The Schedule data set acts as a secondary source of information for all Label data set observations that are linked to it. The priority mechanism is described in the section "Determining the Vertical Position" on page 554.

Determining the Vertical Position

You can specify the vertical position for a label string in one of two ways, either directly by using the _Y variable in the Label data set or indirectly by associating the label with an activity or activities. In the latter case, the vertical position is determined by the relative position of the activity on the activity axis of the Gantt chart.

Directly using _Y

The procedure determines the vertical position using the _Y variable. You specify the coordinate system for the value of _Y with the optional _YSYS variable. A value of DATA or DATAVAL for the _YSYS variable indicates that the unit of measurement is data values. This is also the default coordinate system for _Y. A value of PCT or PCTVAL indicates that the unit of measurement is percentage of the procedure output area. When the coordinate system for _Y is based on data values, the values that _Y can take are restricted to positive real numbers with the exception of -1, which is a special value indicating that the label be displayed for every activity. In effect, this is a more concise way of linking a label to every activity.

Indirectly using LABVAR=

If the _Y variable does not exist or its value is missing, the procedure uses the value of the LABVAR variable to determine the vertical position of the label. If the LABVAR= option is specified and the value of the LABVAR variable is nonmissing, the observation is displayed for every activity that provides a matching value for the LABVAR variable. It is quite possible that there are no activities that provide a match, in which case the Label data set observation is ignored. Likewise, the Label data set observation is ignored if the value of the LABVAR variable is missing,

When the vertical position is based on an integer value for _Y or linkage using the LABVAR variable, the default position for the baseline of the string is the top of the first schedule bar corresponding to the activity (unless offsets _XOFFSET or _YOFFSET are used).

Determining the Horizontal Position

The procedure attempts to determine the horizontal position using the _X variable. You specify the coordinate system for the value of _X with the optional _XSYS variable. A value of DATA or DATAVAL for the _XSYS variable indicates that the unit of measurement is data values. This is also the default coordinate system for

_X. A value of PCT or PCTVAL indicates that the unit of measurement is percentage of the procedure output area.

If the _X variable does not exist or its value is missing, the procedure ignores the Label data set observation if the observation is not linked to an activity in the Schedule data set. However, if the label is linked to an activity (either by the LABVAR variable or a value of -1 for _Y, as described previously), the procedure extracts the horizontal position using the _XVAR variable in the Label data set. The _XVAR variable values are names of numeric variables in the Schedule data set. If the _XVAR value is not missing, the horizontal position is the value of the specified variable in the Schedule data set corresponding to the activity. If no such variable exists in the Schedule data set or its value is missing, no label is displayed for this particular (activity, label) link. As with the _X variable, the _XSYS variable names the unit of measurement for the associated Schedule data set variable.

Coordinate Systems

Coordinates can be specified in data values and percentages. It is important to note a significant difference between these two systems when using multiple pages. A data coordinate value is a point along either the time or activity axis, and it can be related to a page number and to a position on that page in the relevant direction. A percentage value, on the other hand, cannot be related to a particular page and, as such, is treated as applicable to every single page. It is possible to mix data and percentage coordinates. That is, the horizontal position can be in data values and the vertical position can be in percentage values, and vice versa. By mixing coordinate systems, you can get as flexible as you want in labeling Gantt charts.

- If both coordinates are in data values, the label is displayed at a specific coordinate on a specific page.
- If the horizontal coordinate is a percentage, the label is displayed at this horizontal position for every page that corresponds to the vertical position. Likewise, if the vertical position is a percentage, the label is displayed at this vertical position for every page that corresponds to the horizontal position. For example, you can display certain headings at the top of the Gantt chart or at the bottom of the Gantt chart by using a data value for the vertical position and a percentage value for the horizontal position.
- If the horizontal and vertical coordinates are both percentages, the label is displayed on every page at the specified coordinate. This feature can be used to display text that appears on every page, much like titles and footnotes, for example.

Determining the Label String

The technique for determining the label string is similar to that of determining the horizontal position.

As far as possible, the procedure attempts to use the _LABEL variable. If the _LABEL variable does not exist or its value is missing, the procedure ignores the label data observation if the observation is not linked to an activity in the Schedule data set. However, if the label is linked to an activity (either by the LABVAR variable or a value of -1 for _Y, as described previously), the procedure extracts the text string from the Schedule data set using the _LVAR variable. The _LVAR variable values are names of variables in the Schedule data set. If the _LVAR value is not missing, the text string is the value of the specified variable in the Schedule data set corresponding to the activity. If no such variable exists in the Schedule data set or if the value is missing, no label is displayed for this particular (activity, label) link.

Note that the _LABEL variable and the Schedule data set variables named by _LVAR are not restricted to be of character type. These variables can be character or numeric, formatted or unformatted. The strings are displayed using the following rules:

- If the variable is of character type, the label is the character string corresponding to the given activity.
- If the variable is of numeric type and formatted, the label is the formatted string.
- If the variable is of numeric type and unformatted, the label is the number displayed as a string with an integer part of up to LABMAXINT= digits and a maximum of MAXDEC= decimal positions. The LABMAXINT= and MAXDEC= options are specified in the PROC GANTT statement and their default values are 16 and 2, respectively.

Optional Information

In addition to specifying the horizontal and vertical coordinates as described previously, you can also specify a relative offset from these values using the _XOFFSET and _YOFFSET variables. These are optional variables and their default values are both 0. The unit of measurement for the _XOFFSET variable is in MININTERVAL units, and the direction of increase is from left to right. The unit of measurement for the _YOFFSET variable is in barheights, and the direction of increase is from top to bottom. When labels are split, the offset variables pertain only to the first piece of the label. The positions of the remaining split pieces are determined from the positioning of the first piece. The adjusted coordinate after taking the offsets into account is what is used for the placement of the string and is known as the referenced coordinate.

You can control the color and font of the label strings using the _CLABEL and _FLABEL variables, respectively. The values for the _CLABEL variable are any valid SAS/GRAPH color names. If the _CLABEL variable does not exist or its value is missing, the value of the CTEXT= option in the CHART statement is used. The values for the _FLABEL variable are any valid SAS/GRAPH font names. If the _FLABEL variable does not exist or its value is missing, the value of the FONT= option in the CHART statement is used.

You can control the height of the label strings with the _HLABEL variable. The units of measurement are in barheights. If the _HLABEL variable does not exist or its value is missing, the default value of 1 is used.

You can specify the angle of the character baseline with respect to the horizontal in degrees using the _ALABEL variable. If the _ALABEL variable does not exist or its value is missing, the default value of 0 is used. You can specify the rotation angle of each character in the string in degrees with the _RLABEL variable. If the _RLABEL variable does not exist or its value is missing, the default value of 0 is used.

You can control the alignment of the string with the _JLABEL variable. Strings can be displayed left-justified, right-justified, or centered at the specified coordinate. By default, all strings are displayed left-justified. The valid values are L or LEFT for left justification, R or RIGHT for right justification, and C or CENTER for centered justification.

The _PAGEBRK variable gives you displaying control when the referenced coordinate of a label coincides with a pagebreak tickmark and the horizontal coordinate is measured in data values. You can specify on which of the two pages you would like the label to be displayed. The default always displays the label on the first page associated with the common tickmark except when the tickmark is the very first tickmark on the Gantt chart. Valid values are 0 (default), 1 (use first page), or 2 (use second page).

Variables in the LABELDATA= data set

The following table lists all the variables associated with the Label data set and their interpretations by the GANTT procedure. The table also lists for each variable its type, the possible values it can assume, and its default value.

Table 8.9 Label Data Set Variables

Name	Type	Description	Allowed Values	Defaults
_Y	N	y position		
_X	N	x position		
_LABEL	C/N	Label string		
_XVAR	C	Name of numeric SAS var in DATA= ds for x position		
_LVAR	C	Name of SAS var in DATA= ds for label string		
_XSYS	C	Coordinate system for _X, _XVAR	DATA, DATAVAL, PCT, PCTVAL	DATA
_YSYS	C	Coordinate system for _Y	DATA, DATAVAL, PCT, PCTVAL	DATA
_PAGEBRK	N	Resolve pagebreak referenced display	0, 1, 2	0
_XOFFSET	N	Horizontal offset in minintervals		0
_YOFFSET	N	Vertical offset in bar heights		0
_ALABEL	N	Baseline angle in degrees		0
_CLABEL	C	SAS/GRAPH color name		CTEXT=
_FLABEL	C	SAS/GRAPH font name		FONT=
_HLABEL	N	Height in barheights		1
_JLABEL	C	Justify text	L, LEFT, R, RIGHT, C, CENTER	L
_RLABEL	N	Character rotation in degrees		0
LABVAR=	C/N	Variable linking activities to labels		

Web-Enabled Gantt Charts

The WEB variable enables you to define an HTML reference for each activity. This HTML reference is currently associated with all the schedule bars, milestones, and ID variables that correspond to the activity. The WEB variable is a character variable, and the values need to be of the form “HREF=htmlpage.”

In addition, you can also store the coordinate and link information defined by the WEB= option in a SAS data set by specifying the IMAGEMAP= option in the PROC GANTT statement. By processing this SAS data set using a DATA step, you can generate customized HTML pages for your Gantt chart.

Mode-Specific Differences

All the options that are valid for line-printer, full-screen, and graphics mode Gantt charts are explained in detail in the section "Syntax: GANTT Procedure" on page 494. With few exceptions, the options listed in the section "General Options" on page 501 have the same interpretation in all three modes.

Table 8.10 lists those line-printer options that have a different interpretation for the graphics version of PROC GANTT. Table 8.11 lists options specific for graphics charts and the equivalent line-printer/full-screen option. Table 8.12 lists options specific for line-printer and full-screen charts and the equivalent graphics option.

Table 8.10 Line-Printer Options and Corresponding Graphics Interpretation

Line-Printer Option	Graphics Mode Interpretation
SCALE=*scale*	One column is denoted by (1/*h*)% of the screen width, where HPOS=*h*.
SKIP=*skip*	*skip* number of bar heights are skipped between the bars for two consecutive activities. The value 0 is not valid in the graphics case.

Table 8.11 Graphics Mode Options and Line-Printer/Full-Screen Equivalent

Graphics Option/Statement	Line-Printer/Full-Screen Equivalent
LHCON=*linetype*	HCONCHAR=*'character'*
LREF=*linetype*	REFCHAR=*'character'*
LTNOW=*linetype*	TNCHAR=*'character'*
NOFRAME	FORMCHAR=*'string'*
PATTERN statement	JOINCHAR=*'string'* and SYMCHAR=*'string'*
SYMBOL statement	First character of variable name is plotted (See CHART specifications)
VMILE=*value*	MILECHAR=*'character'*
WTNOW=*width*	TNCHAR=*'character'*

Table 8.12 Line-Printer/Full-Screen Mode Specific Options

Line-Printer/Full-Screen Option	Graphics Equivalent
FORMCHAR=*'string'*	NOFRAME
HCONCHAR=*'character'*	LHCON=*linetype*, CHCON=*color*
HOLICHAR=*'character'*	PATTERN statement 7
JOINCHAR=*'string'*	PATTERN statements 1-6, 8, and 9
MILECHAR=*'character'*	VMILE=*value*, FMILE=*font*, HMILE=*height*, CMILE=*color*
REFCHAR=*'character'*	LREF=*linetype*, CREF=*color*
SYMCHAR=*'string'*	PATTERN statements 1-6, 8, and 9
TNCHAR=*'character'*	LTNOW=*linetype*, WTNOW=*width*, CTNOW=*color*

Displayed Output

The GANTT procedure produces one or more pages of displayed values and a plot of the schedule. If the SUMMARY option is specified, the chart is preceded by a detailed description of the symbols used. A legend is displayed at the foot of the chart on each page unless suppressed by the NOLEGEND option. The main body of the output consists of columns of the ID values and the Gantt chart of the schedule.

For each activity in the project, PROC GANTT displays the values of the ID variables in the ID columns and plots any combination of the following schedules: the predicted schedule as specified by the early and late start and finish times, the actual schedule as specified by the actual start and finish times, the resource-constrained schedule as specified by the resource-constrained start and finish times, and the baseline schedule as specified by the baseline start and finish times. The procedure looks for default variable names for each of these times (E_START for early start, E_FINISH for early finish, S_START for resource-constrained start times, and so on), or you can explicitly specify the names of the appropriate variables using the ES=, EF=, LS=, . . . options.

By specifying the COMBINE option in the CHART statement, you can request PROC GANTT to represent early, late, and actual schedule information on a single bar rather than use two separate bars (one for the early and late schedules and the other for the actual schedule.) PROC GANTT automatically draws a timenow line when the COMBINE option is specified with the property that all times to the left of the line represent the actual schedule times (that is, times that have already taken place) and all times to the right of the line represent the predicted early/late schedule times (times that have not yet taken place.)

Normally, each observation in the Schedule data set is assumed to denote a new activity, and a new set of ID values are displayed and the schedules corresponding to this activity are plotted on the chart. There are two exceptions to this rule:

- If the ID values for two or more consecutive observations are identical, only the first such observation is used.
- If there is a variable named SEGMT_NO in the Schedule data set, PROC GANTT assumes that the data set contains observations for segments of activities that were split during resource-constrained scheduling. In accordance with the conventions used by PROC CPM, only observations with a missing value for SEGMT_NO are assumed to denote a new activity. Further, the data are assumed to be sorted by SEGMT_NO for each activity. For each activity, PROC GANTT plots the schedules corresponding to the ES, EF, LS, LF, AS, and AF variables on the basis of the first observation for this activity, namely the observation with a missing value for the SEGMT_NO variable. This observation is also the one used for displaying values for the ID variables for this activity. If the activity is not split, this same observation is also the one used to plot the resource-constrained schedule as well as the baseline schedule. However, if the activity is split, then all the observations for this activity with integer values for the variable SEGMT_NO are used to plot the resource-constrained schedule as disjoint segments on the line used for plotting the S_START and S_FINISH times. Furthermore, PROC GANTT plots the baseline schedule corresponding to the BS and BF variables based on the last such observation, namely the observation with the largest value for the SEGMT_NO variable.

In addition to the schedules that are plotted, the Gantt chart also displays any variables specified in the CHART statement. Holidays, weekends, and breaks within a day are marked as appropriate. For details on how to specify holidays, weekends, and breaks within a day, see the section "Multiple Calendars and

Holidays" on page 531. You can also represent zero duration activities with milestone symbols, draw a timenow line to reflect the current time of the project, draw horizontal connect lines, draw vertical reference lines, and group the activities by zones on the Gantt chart. It is important to note that all times are plotted at the start of the appropriate time period. Thus, if the chart starts on June 1, 2004, in column 15 of the page and the value of E_START is '2JUN04,' MININTERVAL=DAY, and SCALE=5, then the early start time is plotted in column 20.

Each activity is identified by a job number (unless the NOJOBNUM option is used), which appears as the first column of activity text. The next column of activity text identifies the values of the ZONE= variable, if specified. This column can be suppressed by specifying the NOZONECOL option in the CHART statement. Next to appear are the ID variables in the order in which they are specified in the CHART statement. If the time axis of the chart is very wide, causing it to be divided across more than one page, the ID variables, by default, do not appear on continuation pages. You need to specify the IDPAGES option to produce the ID variable columns on every page. By default, if the ID variables occupy too much space, leaving no room for the chart to be started on the first page, they are omitted and a warning message is printed to the log. You can override this behavior by using the MAXIDS option. Column headings for the ZONE and ID variables consist of either variable labels (if they are present and if space permits) or variable names. To suppress variable labels in column headings, use the NOLABEL system option. If a ZONE or ID variable is formatted, the value is displayed using that format. If the CRITFLAG option is specified, a flag is displayed to the right of the ID values that indicates how critical the activity is. This flag is also repeated on continuation pages if the time axis occupies more than one page. The body of the chart starts to the right of this flag.

By default, the GANTT procedure is invoked in graphics mode. In graphics mode, you can fit the Gantt chart entirely on one page by specifying the COMPRESS option in the CHART statement. The HPAGES= and VPAGES= options take this one step further by enabling you to control the number of pages that you want the Gantt chart to be compressed into. The PCOMPRESS option behaves much like the COMPRESS option except that all compression is performed in a proportional manner, that is, by maintaining the aspect ratio of the Gantt chart.

PROC GANTT can display the precedence relationships (including nonstandard types) between activities on the Gantt chart by means of directed links between activities. Each link is drawn so as to convey the type of precedence relationship it represents. See the section "Specifying the Logic Options" on page 546 for a detailed description on how this can be done.

In addition, graphics mode provides you with the easy-to-use automatic text annotation facility to generate labels on the Gantt chart. You can link labels and their coordinates to variables in the schedule data set and also have complete control over all attributes such as font, color, angle, rotation, and so forth. You also have the additional capability of annotating text and graphics independently on the Gantt chart by using the SAS/GRAPH Annotate facility.

The GANTT procedure offers you a wide variety of options in addition to text, bar, symbol, and line formatting controls to customize your Gantt chart. These features enable you to create a wide variety of charts such as Logic Gantt charts, zoned Gantt charts, multiproject Gantt charts, Web-enabled Gantt charts, and multiprocess Gantt charts, to name but a few.

Macro Variable _ORGANTT

The GANTT procedure defines a macro variable named _ORGANTT, which is set at procedure termination. This variable contains a character string that indicates the status of the procedure and also provides chart specific information with respect to each Gantt chart produced by invocation of the GANTT procedure. This includes charts resulting from multiple CHART statements and BY groups.

The format of the _ORGANTT string for a GANTT procedure invocation with *n* CHART statements is as follows:

STATUS= REASON= CHART1 chart1info # . . . CHART*n* chart*n*info #

where the value of STATUS= is either SUCCESSFUL or ERROR_EXIT, and the value of REASON= is one of the following:

- BADDATA_ERROR
- MEMORY_ERROR
- IO_ERROR
- SEMANTIC_ERROR
- SYNTAX_ERROR
- GANTT_BUG
- UNKNOWN_ERROR

The notation chart*i*info is a string of the form

SCALE= INCREMENT= SKIP= HPAGES= VPAGES= SEGNAME=

if there are no BY groups, and it is a string of the form

BY1 by1info: . . . BY*m* by*m*info:

where by*j*info is a string of the form

SCALE= INCREMENT= SKIP= HPAGES= VPAGES= SEGNAME=

if there are *m* BY groups. In other words, the macro contains an informational substring for every chart produced, using the symbol "#" as a CHART statement delimiter and the symbol ":" as a BY statement delimiter within CHART statements.

The chart specific information given in _ORGANTT is described below along with the identifying keyword preceding it. It should be noted that these values refer to those actually used in producing the chart and are not necessarily the same as those specified in the invocation of the procedure.

- SCALE= The value of scale
- INCREMENT= The value of increment
- SKIP= The value of skip
- HPAGES= The number of horizontal pages
- VPAGES= The number of vertical pages
- SEGNAME= The name of the first chart segment in graphics mode

NOTE: Some of the information might be redundant or predictable in certain display modes. For example, the value of SEGNAME= is empty in line-printer and full-screen modes. The values of HPAGES= and VPAGES= are equal to 1 in full-screen mode.

This information can be used when PROC GANTT is one step in a larger program that needs to determine whether the procedure terminated successfully or not. Because _ORGANTT is a standard SAS macro variable, it can be used in the ways that all macro variables can be used.

Computer Resource Requirements

There is no inherent limit on the size of the project that the GANTT procedure can accommodate. The number of activities in the Gantt chart is restricted only by the amount of memory available. Other memory-dependent factors are the type of Gantt chart required and the desired display mode.

Naturally, there needs to be a sufficient amount of core memory available in order to invoke and initialize the SAS System as well as to meet the memory requirements of the specific mode in which you invoke the procedure. For example, more memory is required when you use high-resolution graphics than when you use line-printer mode because the graphics sublibrary has to be loaded. The procedure attempts to store all the data in core memory. However, if there is insufficient core memory available for the entire project, the GANTT procedure resorts to using utility data sets and swaps between core memory and utility data sets as necessary.

The data storage requirement for the GANTT procedure is proportional to the number of activities in the project, and it depends on the number of schedule variables, the number of ID variables, and whether the Logic and Labeling options have been specified or not.

ODS Style Templates

ODS style templates, or *styles*, control the overall look of your output. An ODS style template consists of a set of *style elements*. A style element is a collection of *style attributes* that apply to a particular feature or aspect of the output. You can specify a value for each attribute in a style. See Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*), for a thorough discussion of ODS Graphics.

To create your own style or to modify a style for use with ODS Graphics, you need to understand the relationships between style elements and graph features. This information is provided in the ODS Graphics documentation at **`http://support.sas.com/documentation/onlinedoc/base/`**. You can create and modify style templates with the TEMPLATE procedure. For more information, see the section "TEMPLATE Procedure: Creating a Style Template" in the *SAS Output Delivery System: User's Guide*. Kuhfeld (2010) also offers detailed information and examples.

PROC GANTT Style Template

A predefined ODS style template named GANTT is available for the GANTT procedure. You can use the template to maintain a consistent appearance in all graphical output produced by the procedure.

To change the current style, specify the STYLE= option in an ODS destination statement. The specified style is applied to all output for that destination until you change or close the destination or start a new SAS session. For example, the following statement specifies that ODS should apply the GANTT style template to all HTML output:

```
ods html style=gantt;
```

To disable the use of graphical styles, specify the SAS system option NOGSTYLE.

The parent style template for the GANTT style is the DEFAULT style. Table 8.13 lists the style elements (in bold) and corresponding attributes specified in the GANTT style. The table also indicates which (if any) PROC GANTT options or graphics options (in a GOPTIONS statement) can be used to override the value of a style attribute.

Table 8.13 Style Elements and Attributes in the GANTT Style

Element/Attributes	Description	GANTT Option	GOPTION
GraphColors	Colors of various graph features	PATTERN=	CPATTERN=, COLORS=
gdata1	Duration of a noncritical activity	PATTERN=	CPATTERN=, COLORS=
gdata2	Slack time for a noncritical activity	PATTERN=	CPATTERN=, COLORS=
gdata3	Duration of a critical activity	PATTERN=	CPATTERN=, COLORS=
gdata4	Slack time for a supercritical activity	PATTERN=	CPATTERN=, COLORS=
gdata5	Duration of a supercritical activity	PATTERN=	CPATTERN=, COLORS=
gdata6	Actual duration of an activity	PATTERN=	CPATTERN=, COLORS=
gdata7	Break due to a holiday	PATTERN=	CPATTERN=, COLORS=
gdata8	Resource-constrained duration of an activity	PATTERN=	CPATTERN=, COLORS=
gdata9	Baseline duration of an activity	PATTERN=	CPATTERN=, COLORS=
gaxis	Axis	CAXIS=	COLORS=
ggrid	Horizontal connecting lines, zone lines	CHCON=, CZONE=	COLORS=
gdata	Default		COLORS=
gcdata	Precedence connections	CPREC=	COLORS=
greferencelines	Reference and timenow lines	CREF=, CTNOW=	COLORS=
gtextt	Title text		CTITLE=
gtext	Text		CTEXT=
glabel	Labels		COLORS=

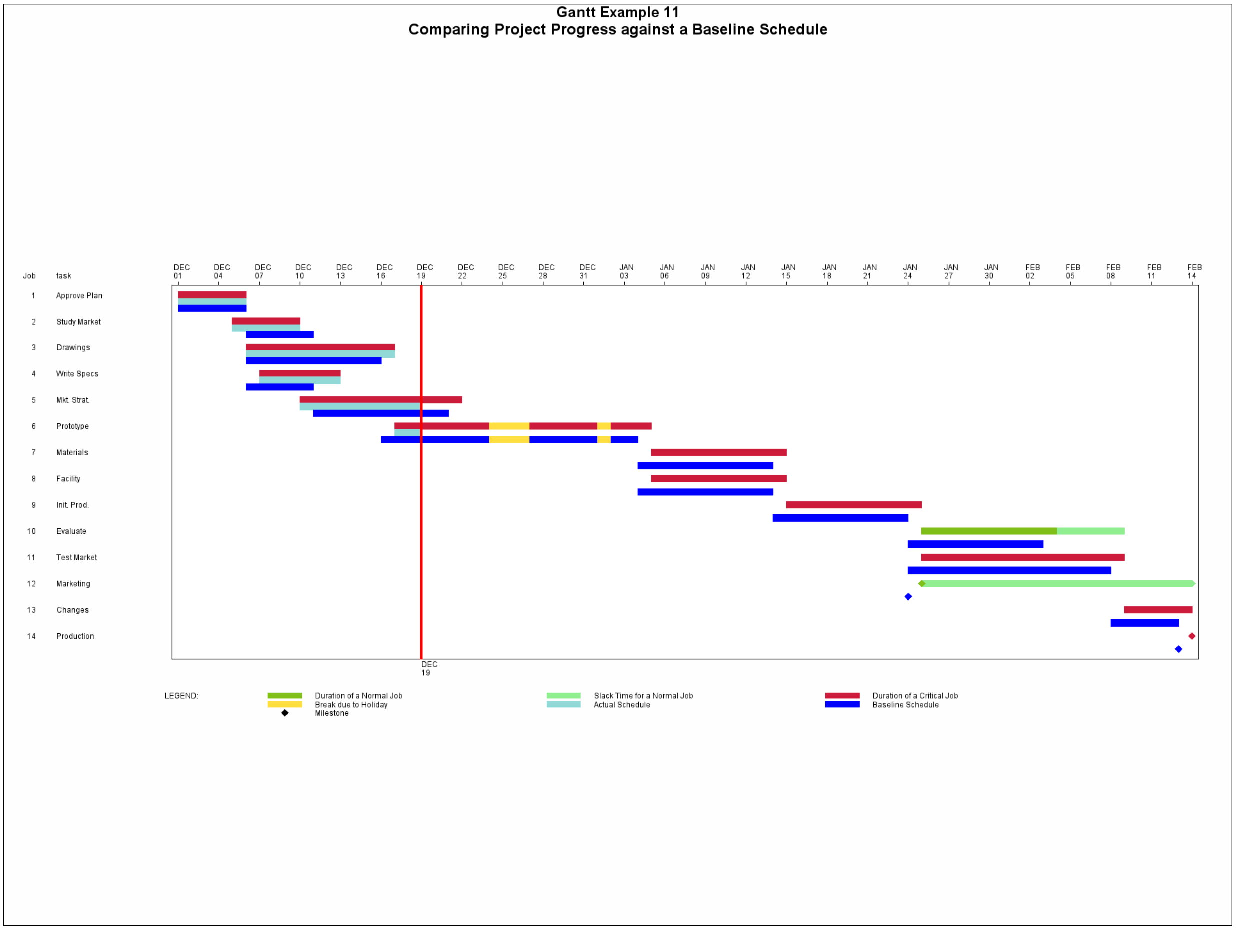
Table 8.13 (continued)

Element/Attributes	Description	GANTT Option	GOPTION
GraphFonts	Fonts for various graph features		
GraphDataFont	Default		FONT=
GraphLabelFont	Labels		FONT=
GraphTitleFont	Title text		FTITLE=
GraphAxisLines	Attributes related to graph axes		
Color	GraphColors('gaxis')	CAXIS=	COLORS=
GraphGridLines	Attributes related to horizontal connecting lines and zone lines		
Color	GraphColors('ggrid')	CHCON=, CZONE=	COLORS=
GraphConnectLine	Attributes related to precedence connections		
Color	GraphColors('gcdata')	CPREC=	COLORS=
GraphReference	Attributes related to reference and timenow lines		
Color	GraphColors('greferencelines')	CREF=, CTNOW=	COLORS=
GraphDataText	Attributes related to general text		
Color	GraphColors('gtext')	CTEXT=	COLORS=
Font	GraphFonts('GraphDataFont')	FONT=	FONT=
GraphTitleText	Attributes related to title text		
Color	GraphColors('gtextt')		CTITLE=
Font	GraphFonts('GraphTitleFont')		FTITLE=
GraphTitle1Text	Attributes related to the first title text		
Color	GraphColors('gtextt')		CTITLE=
Font	GraphFonts('GraphTitleFont')		FTITLE=
GraphLabelText	Attributes related to label text		
Color	GraphColors('glabel')	_CLABEL variable in the LABDATA= data set	COLORS=
Font	GraphFonts('GraphLabelFont')	_FLABEL variable in the LABDATA= data set	FONT=
GraphDataDefault	Default values for the attributes specified in Table 8.14		
Color	GraphColors('gdata')		COLORS=

Attributes that you do not override retain the values specified in the style template.

Figure 8.15 demonstrates features of the GANTT graphical style. The GANTT chart in the figure is the first output from Example 8.11.

Figure 8.15 GANTT Style Template: Example



Default Values

If the SAS system option GSTYLE is in effect (this is the default), then the default values of certain PROC GANTT options can depend on the current ODS style template. Table 8.14 lists these PROC GANTT options and lists the order in which PROC GANTT searches for each option's default value. The order assumes that the GSTYLE system option is in effect; if that is not the case, then the steps that refer to ODS style templates are ignored. Names with arguments indicate style elements and attributes of the current ODS style template. For example, "GraphAxisLines('Color')" refers to the Color attribute of the GraphAxisLines element.

Table 8.14 PROC GANTT Options: Search Orders for Default Values

Option	Search Order for Default Color
CAXIS=	1. The Color attribute of the GraphAxisLines element of the current ODS style template 2. The Color attribute of the GraphDataDefault element of the current ODS style template 3. The first color in the COLORS= list in the GOPTIONS statement
CFRAME=	1. No color filling the axis area (if the GSTYLE system option is not in effect)

Table 8.14 (continued)

Option	Search Order for Default Color
	2. The Color attribute of the GraphWalls element of the current ODS style template 3. No color filling the axis area
CHCON=	1. The Color attribute of the GraphGridLines element of the current ODS style template 2. The Color attribute of the GraphDataDefault element of the current ODS style template 3. The first color in the COLORS= list in the GOPTIONS statement
CPREC=	1. The Color attribute of the GraphConnectLine element of the current ODS style template 2. The Color attribute of the GraphDataDefault element of the current ODS style template 3. The first color in the COLORS= list in the GOPTIONS statement
CREF=	1. The Color attribute of the GraphReference element of the current ODS style template 2. The Color attribute of the GraphDataDefault element of the current ODS style template 3. The first color in the COLORS= list in the GOPTIONS statement
CTEXT=	1. The value specified for the CTEXT= option in the GOPTIONS statement 2. The Color attribute of the GraphDataText element of the current ODS style template 3. The Color attribute of the GraphDataDefault element of the current ODS style template 4. The first color in the COLORS= list in the GOPTIONS statement
CTNOW=	1. The Color attribute of the GraphReference element of the current ODS style template 2. The Color attribute of the GraphDataDefault element of the current ODS style template 3. The first color in the COLORS= list in the GOPTIONS statement
CZONE=	1. The Color attribute of the GraphGridLines element of the current ODS style template 2. The Color attribute of the GraphDataDefault element of the current ODS style template 3. The first color in the COLORS= list in the GOPTIONS statement
FONT=	1. The value specified for the FTEXT= option in the GOPTIONS statement 2. The Font attribute of the GraphDataText element of the current ODS style template 3. The default hardware font for the graphics output device

Examples: GANTT Procedure

This section contains examples that illustrate several of the options and statements available with PROC GANTT in the different display modes. Example 8.1 and Example 8.2 illustrate the GANTT procedure in line-printer mode, and Example 8.3 through Example 8.27 illustrate the GANTT procedure in graphics mode.

Line-Printer Examples

Example 8.1 shows how to obtain a basic line-printer Gantt chart using the default options. Example 8.2 demonstrates how to use various options to customize the Gantt chart for the same project.

Example 8.1: Printing a Gantt Chart

This example shows how to use the GANTT procedure to obtain a basic line-printer Gantt chart using the default options. The following data describe the precedence relationships among the tasks involved in the construction of a typical floor in a multistory building. The first step saves the precedence relationships in a SAS data set. The variable ACTIVITY names each task, the variable DUR specifies the time it takes to complete the task in days, and the variables SUCCESS1 to SUCCESS4 specify tasks that are immediate successors to the task identified by the ACTIVITY variable.

PROC CPM determines the shortest schedule for the project that finishes before September 1, 2004. The solution schedule, saved in a SAS data set, is next sorted by the early start time before invoking the GANTT procedure to plot the schedule. Since the DATA= option is not specified, PROC GANTT uses the sorted data set to produce the schedule since it is the most recently created data set. The Gantt chart in Output 8.1.1 is plotted on two pages because there are too many observations (29) to fit on one page. Note that the observations are split into two groups containing 15 and 14 observations, respectively, so that the chart size on each page is approximately equal. The time axis is labeled from June 21, 2004, to September 1, 2004, since these are the minimum and maximum dates in the Schedule data set. A legend is displayed at the bottom of the chart on each page.

```
title 'Gantt Example 1';
title2 'Printing a Gantt Chart';

data;
   format activity $20. success1 $20. success2 $20. success3 $20.
                        success4 $20.;
   input activity dur success1-success4;
   datalines;
form                 4 pour . . .
pour                 2 core . . .
core                14 strip spray_fireproof insulate_walls .
strip                2 plumbing curtain_wall risers doors
strip                2 electrical_walls balance_elevator . .
curtain_wall         5 glaze_sash . . .
glaze_sash           5 spray_fireproof insulate_walls . .
spray_fireproof      5 ceil_ducts_fixture . . .
```

```
ceil_ducts_fixture   5 test . . .
plumbing            10 test . . .
test                 3 insulate_mechanical . . .
insulate_mechanical  3 lath . . .
insulate_walls       5 lath . . .
risers              10 ceil_ducts_fixture . . .
doors                1 port_masonry . . .
port_masonry         2 lath finish_masonry . .
electrical_walls    16 lath . . .
balance_elevator     3 finish_masonry . . .
finish_masonry       3 plaster marble_work . .
lath                 3 plaster marble_work . .
plaster              5 floor_finish tiling acoustic_tiles .
marble_work          3 acoustic_tiles . . .
acoustic_tiles       5 paint finish_mechanical . .
tiling               3 paint finish_mechanical . .
floor_finish         5 paint finish_mechanical . .
paint                5 finish_paint . . .
finish_mechanical    5 finish_paint . . .
finish_paint         2 caulking_cleanup . . .
caulking_cleanup     4 finished . . .
;

* invoke cpm to find the optimal schedule;

proc cpm finishbefore date='1sep04'd;
   activity activity;
   duration dur;
   successors success1-success4;
run;

* sort the schedule by the early start date;

proc sort;
   by e_start;
run;

* invoke proc gantt to print the schedule;

proc gantt lineprinter;
run;
```

Output 8.1.1 Printing a Gantt Chart

Gantt Example 1
Printing a Gantt Chart

```
         JUN   JUN   JUL   JUL   JUL   JUL   JUL   AUG   AUG   AUG   AUG   AUG   SEP
  Job    21 24 27 30 03 06 09 12 15 18 21 24 27 30 02 05 08 11 14 17 20 23 26 29 01
        -+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+-
        |                                                                         |
    1   |*===*                                                                    |
        |                                                                         |
    2   |    *=*                                                                  |
        |                                                                         |
    3   |      *=============*                                                    |
        |                                                                         |
    4   |                    *=*                                                  |
        |                                                                         |
    5   |                    *=*                                                  |
        |                                                                         |
    6   |                      *====*                                             |
        |                                                                         |
    7   |                      <---------*.........>                              |
        |                                                                         |
    8   |                      <----<---->....>                                   |
        |                                                                         |
    9   |                      <>.....................<>                          |
        |                                                                         |
   10   |                      <---------<----->.........>                        |
        |                                                                         |
   11   |                      <-->...................<..>                        |
        |                                                                         |
   12   |                       <->....................<.>                        |
        |                                                                         |
   13   |                         <-->...................<..>                     |
        |                                                                         |
   14   |                           *====*                                        |
        |                                                                         |
   15   |                                *====*                                   |
        |                                                                         |
        -+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+-


                                       LEGEND


                       Symbol     Explanation

                       <---->     Duration of a Normal Job

                       >....>     Slack Time for a Normal Job

                       *====*     Duration of a Critical Job
```

Output 8.1.1 *continued*

Gantt Example 1
Printing a Gantt Chart

```
         JUN   JUN   JUL   JUL   JUL   JUL   JUL   AUG   AUG   AUG   AUG   AUG   SEP
Job      21 24 27 30 03 06 09 12 15 18 21 24 27 30 02 05 08 11 14 17 20 23 26 29 01
       -+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+-
       |                                                                         |
 16    |                                <---->.....<....>                        |
       |                                                                         |
 17    |                                     *====*                              |
       |                                                                         |
 18    |                                          *==*                           |
       |                                                                         |
 19    |                                             *==*                        |
       |                                                                         |
 20    |                                                *==*                     |
       |                                                                         |
 21    |                                                   *====*                |
       |                                                                         |
 22    |                                                   <-<>.>                |
       |                                                                         |
 23    |                                                        *====*           |
       |                                                                         |
 24    |                                                        <-<>.>           |
       |                                                                         |
 25    |                                                        *====*           |
       |                                                                         |
 26    |                                                             *====*      |
       |                                                                         |
 27    |                                                             *====*      |
       |                                                                         |
 28    |                                                                  *=*    |
       |                                                                         |
 29    |                                                                    *===*|
       |                                                                         |
       -+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+-


                                   LEGEND


                    Symbol     Explanation

                    <---->     Duration of a Normal Job

                    >....>     Slack Time for a Normal Job

                    *====*     Duration of a Critical Job
```

Example 8.2: Customizing the Gantt Chart

This example shows how to control the format of the Gantt chart using CHART statement options. The Schedule data set used by PROC GANTT is the same as that used in Example 8.1. Output 8.2.1 is on three pages; the first page contains a detailed description of the various symbols used by the procedure to plot the schedule. This description is produced by using the SUMMARY option. The next two pages contain the Gantt chart. The LINEPRINTER option invokes the procedure in line-printer mode. The FILL option causes the first page to be filled as completely as possible before the second page is started. Thus, the first page of the chart contains 20 activities while the second page contains only 8 activities.

The MININTERVAL=WEEK specification defines the units of time for axis labeling. The SCALE=5 specification causes five columns of the chart to be used to display one week. The SKIP=2 specification causes two lines to be skipped between observations. The NOLEGEND option suppresses displaying of the legend, while the NOJOBNUM option causes job numbers to be omitted. The CRITFLAG option is used to produce the flag to the left of the main chart indicating if an activity is critical. Specifying BETWEEN=2 sets the number of columns between consecutive ID columns equal to 2. The REF= option produces the reference lines shown on the chart on the specified dates. The ID statement is used to display the activity names to the left of the chart. The ID statement also causes the activity 'strip' to appear only once in the chart. Thus, there are only 28 activities in this chart instead of 29, as in Example 8.1.

```
title 'Gantt Example 2';
title2 'Customizing the Gantt Chart';

proc gantt lineprinter;
   chart / summary
           fill
           mininterval=week scale=5
           skip=2
           nolegend
           nojobnum critflag between=2
           ref='10jun04'd to '30aug04'd by 15;
   id activity;
   run;
```

Output 8.2.1 Customizing the Gantt Chart

Gantt Example 2
Customizing the Gantt Chart

```
                    Summary


 Symbols used for different times on the schedule

 Variable      Symbol          Variable      Symbol


 E_START         <             L_START         <
 E_FINISH        >             L_FINISH        >


              Miscellaneous Symbols

Symbol      Explanation

  |         Reference Line

  *         Overprint character when start or
            finish times coincide

Symbols used for joining start and/or finish times

Symbol      Explanation

    -       Duration of non-critical job

    .       Slack time for non-critical job

    =       Duration of critical job

    -       Slack time(neg.) for supercritical job

    *       Duration of supercritical job
```

Output 8.2.1 *continued*

Gantt Example 2
Customizing the Gantt Chart

```
                         Summary (Contd.)


                  Some examples of typical strings

         String             Description

         <--->...<...>      Duration followed by slack time:
                            early finish before late start

         <---<--->...>      Duration followed by slack time:
                            early finish after late start

         <---*...>          Duration followed by slack time:
                            early finish equals late start

         *===*              Duration of job on critical path

         <--->---<***>      Duration preceded by negative slack
                            time for a supercritical job:
                            late finish before early start

         <---<***>***>      Duration preceded by negative slack
                            time for a supercritical job:
                            late finish after early start

         <---****>          Duration preceded by negative slack
                            time for a supercritical job:
                            late finish equals early start
```

Output 8.2.1 *continued*

Gantt Example 2
Customizing the Gantt Chart

```
                            JUN  JUN  JUL  JUL  JUL  JUL  AUG  AUG  AUG  AUG  AUG  SEP
activity             Flag   21   28   05   12   19   26   02   09   16   23   30   06
                           -+----+----+----+----+----+----+----+----+----+----+----+-
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
form                  CR   |*==*          |         |          |          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
pour                  CR   |   **         |         |          |          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
core                  CR   |   |*=========*         |          |          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
strip                 CR   |   |          *=*       |          |          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
curtain_wall          CR   |   |          | *==*    |          |          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
plumbing                   |   |          | <------*|.....>    |          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
risers                     |   |          | <--<--->|.>        |          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
doors                      |   |          | *.......|.......<> |          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
electrical_walls           |   |          | <------<|-->......>|          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
balance_elevator           |   |          | <->.....|.......<.>|          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
port_masonry               |   |          | <->.....|........<>|          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
finish_masonry             |   |          |   <->...|.........<|>         |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
glaze_sash            CR   |   |          |    *===*|          |          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
spray_fireproof       CR   |   |          |        *|=*        |          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
insulate_walls             |   |          |        <|->....<..>|          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
                           -+----+----+----+----+----+----+----+----+----+----+----+-
```

Output 8.2.1 *continued*

Gantt Example 2
Customizing the Gantt Chart

```
                            JUN  JUN  JUL  JUL  JUL  JUL  AUG  AUG  AUG  AUG  AUG  SEP
activity             Flag   21   28   05   12   19   26   02   09   16   23   30   06
                           -+----+----+----+----+----+----+----+----+----+----+----+-
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
ceil_ducts_fixture    CR   |   |          |         | *===*    |          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
test                  CR   |   |          |         |     *=*  |          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
insulate_mechanical   CR   |   |          |         |       *=*|          |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
lath                  CR   |   |          |         |         *|*         |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
plaster               CR   |   |          |         |          |*===*     |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
marble_work                |   |          |         |          |<-<>>     |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
acoustic_tiles        CR   |   |          |         |          |    *===* |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
tiling                     |   |          |         |          |    <<>.> |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
floor_finish          CR   |   |          |         |          |    *===* |         |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
paint                 CR   |   |          |         |          |        *=|*        |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
finish_mechanical     CR   |   |          |         |          |        *=|*        |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
finish_paint          CR   |   |          |         |          |          |*=*      |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
caulking_cleanup      CR   |   |          |         |          |          |  *=*    |
                           |   |          |         |          |          |         |
                           |   |          |         |          |          |         |
                           -+----+----+----+----+----+----+----+----+----+----+----+-
```

Graphics Examples

The following examples illustrate the use of graphics options and the use of PATTERN and SYMBOL statements to produce high resolution graphics quality Gantt charts. In Example 8.3, an extra input data set containing the holiday information is used to mark the holidays used in computing the schedule by PROC CPM. Example 8.4 illustrates the use of the CHART statement to specify milestones and additional variables to be plotted on the chart. Example 8.5 illustrates the use of the COMPRESS option to fit an entire Gantt chart on one page. Example 8.6 illustrates the use of the MININTERVAL= and SCALE= options to control the width of the chart; this example also shows how the chart is divided and continued on the succeeding page when the time axis extends beyond one page. In Example 8.7, the MINDATE= and MAXDATE= options are used to permit viewing of only a portion of the schedule in greater detail. Example 8.8 uses the HOLIDUR= option in conjunction with the INTERVAL= option to mark holidays of varying lengths on the Gantt chart. Example 8.9 illustrates the use of the CALENDAR and WORKDAY data sets to mark holiday information from different calendars on the chart.

In Example 8.10, the actual schedule for each activity is plotted on a separate line in addition to the early and late schedules. Example 8.11 illustrates tracking a project and comparing its progress against a baseline schedule. In Example 8.12, the COMBINE option is used to concatenate the early, late, and actual schedules of a project in progress to produce a single concise schedule that retains all of the vital information of the former schedules. Example 8.13 shows the resource-constrained schedule containing split segments of activities. The ability to bypass the project scheduler, PROC CPM, and directly specify the schedule information to PROC GANTT is demonstrated in Example 8.14. Example 8.15 illustrates the use of the BY statement to obtain Gantt charts for different projects in a multiproject environment. In Example 8.16, the GANTT procedure is used after some data manipulation steps to produce Gantt charts for individuals, each working on different subsets of activities in the project.

In Example 8.17, the HEIGHT= and HTOFF= options are used to modify the text height in relation to the height of the activity bars. The next three examples show you how to invoke the different logic options in order to draw a Logic Gantt chart that displays the precedence relationships between activities. Example 8.18 illustrates use of the ACTIVITY= and SUCCESSOR= options to specify the precedence information in AON format and the LEVEL= option to specify the bar type for the connections. In Example 8.19, the routing control options MAXDISLV=, MAXOFFGV=, MAXOFFLV=, and MININTGV= are used in connection with a project that is specified in AOA format using the TAIL= and HEAD= options in the CHART statement. Example 8.20 demonstrates the specification of nonstandard lag types using the LAG= option in the CHART statement. This example also illustrates use of the PRECDATA= option in the PROC GANTT statement. In Example 8.21, the ANNOTATE= option is used to add graphics and text on a Gantt chart. Example 8.22 illustrates the Automatic Text Annotation facility to label the Gantt chart independently of the SAS/GRAPH Annotate facility. In Example 8.23 a PATTERN variable and a Label data set are used to generate Gantt charts for multiprojects. A very useful chart in project management and multiprocess environments is the multisegment Gantt chart. Example 8.24 illustrates the use of the SEGMT_NO variable and the PATTERN variable to produce a versatile multisegment Gantt chart. In Example 8.25 the ZONE= option is used to produce a zoned Gantt chart. Example 8.26 shows you how to produce a “Web-enabled” Gantt chart that you can use to drill-down your project. Finally, Example 8.27 uses the CHARTWIDTH= option to produce Gantt charts that are consistent in appearance.

In all the examples presented, the early and late schedules are specified in the data set by means of the variables E_START, E_FINISH, L_START, and L_FINISH; hence, the ES=, EF=, LS=, and LF= options are not needed in the CHART statement. Unless otherwise specified, the pattern statements used in the examples are as follows:

```
pattern1 c=green v=s;    /* duration of a non-critical activity  */
pattern2 c=green v=e;    /* slack time for a noncrit. activity   */
pattern3 c=red v=s;      /* duration of a critical activity      */
pattern4 c=magenta v=e;  /* slack time for a supercrit. activity */
pattern5 c=magenta v=s;  /* duration of a supercrit. activity    */
pattern6 c=cyan v=s;     /* actual duration of an activity       */
pattern7 c=black v=e;    /* break due to a holiday               */
pattern8 c=blue v=s;     /* resource schedule of activity        */
pattern9 c=brown v=s;    /* baseline schedule of activity        */
```

Example 8.3: Marking Holidays

This example uses the widget manufacturing project introduced in Chapter 4, “The CPM Procedure.” The data sets used in this example are the same as those used in Example 4.8 to illustrate holiday processing in PROC CPM. The WIDGET data set describes the project in AON format. The variable TASK identifies the activity and the variables SUCC1, SUCC2, and SUCC3 identify the successors to TASK. The variable DAYS defines the duration of an activity. Another data set, HOLIDAYS, defines the holidays that need to be taken into account when scheduling the project. Although the HOLIDAYS data set contains three variables HOLIDAY, HOLIFIN, and HOLIDUR, the HOLIDUR variable is not used in this example. Thus, the Christmas holiday starts on December 24, 2003, and finishes on December 26, 2003. PROC CPM schedules the project to start on December 1, 2003, and saves the schedule in a data set named SAVEH. This data set is shown in Output 8.3.1.

Next, the GANTT procedure is invoked with the specification of HOLIDATA= HOLIDAYS in the PROC GANTT statement and the HOLIDAY= and HOLIEND= options in the CHART statement, causing the Christmas and New Year holidays to be marked on the chart. The resulting Gantt chart is shown in Output 8.3.2. Note that the procedure marks the duration of the holiday with the pattern corresponding to the seventh PATTERN statement. (See the section “Graphics Examples” on page 576 for a list of the pattern statements used in the examples.)

```
options ps=60 ls=80;

title h=2 'Gantt Example 3';
title2 'Marking Holidays';

/* Activity-on-Node representation of the project */
data widget;
   format task $12. succ1-succ3 $12. ;
   input task & days succ1 & succ2 & succ3 & ;
   datalines;
Approve Plan   5  Drawings      Study Market  Write Specs
Drawings      10  Prototype     .             .
Study Market   5  Mkt. Strat.   .             .
Write Specs    5  Prototype     .             .
Prototype     15  Materials     Facility      .
Mkt. Strat.   10  Test Market   Marketing     .
Materials     10  Init. Prod.   .             .
Facility      10  Init. Prod.   .             .
Init. Prod.   10  Test Market   Marketing     Evaluate
Evaluate      10  Changes       .             .
Test Market   15  Changes       .             .
Changes        5  Production    .             .
Production     0  .             .             .
Marketing      0  .             .             .
;

data holidays;
   format holiday holifin date7.;
   input holiday & date7. holifin & date7. holidur;
   datalines;
24dec03  26dec03  4
01jan04  .        .
;

* schedule the project subject to holidays;
proc cpm data=widget holidata=holidays
         out=saveh date='1dec03'd ;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   run;

* sort the schedule by the early start date ;
proc sort;
   by e_start;
   run;

* print the schedule;
proc print data=saveh;
   var task days e_start e_finish l_start l_finish
       t_float f_float;
   run;
```

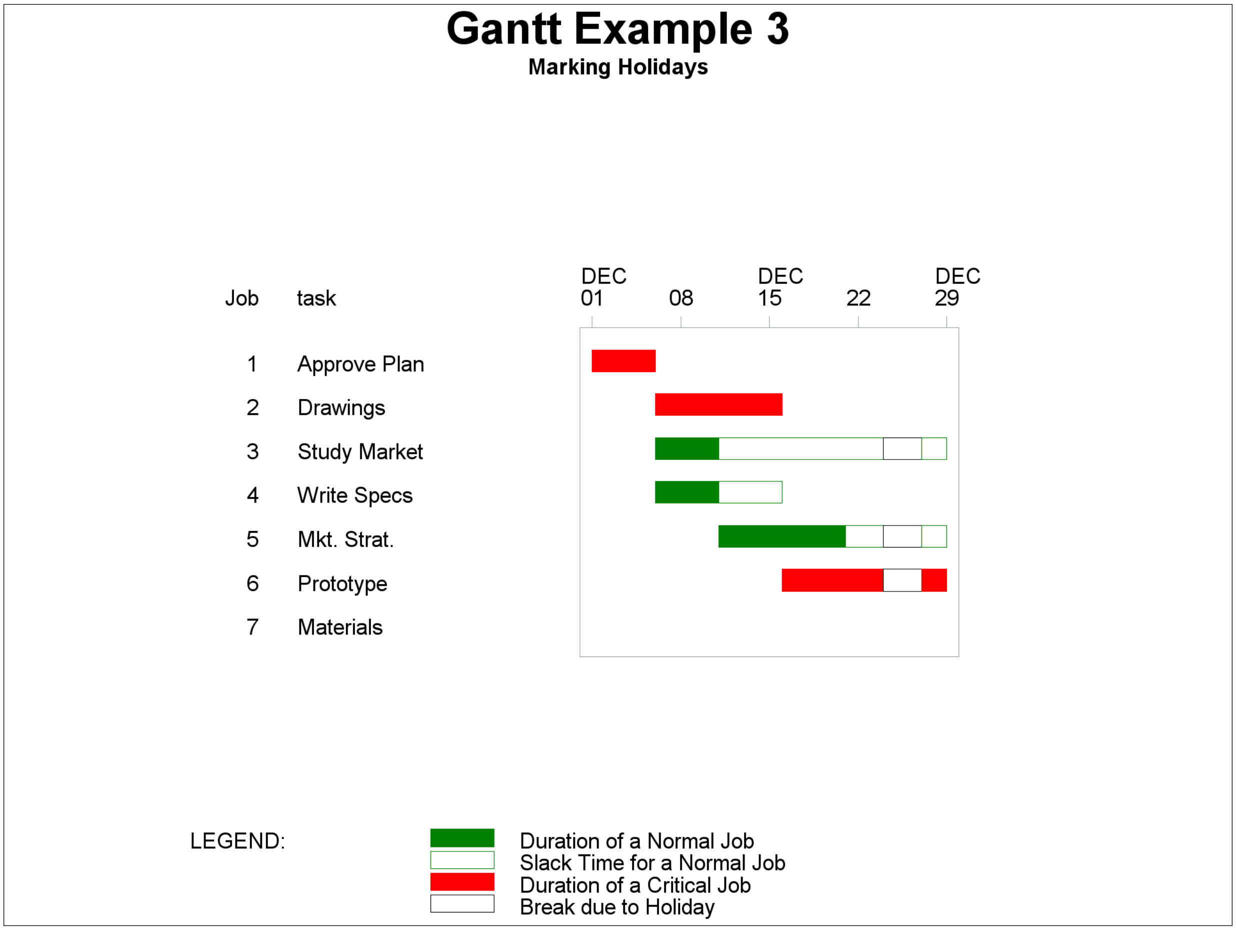
Output 8.3.1 Schedule Data Set SAVEH

Gantt Example 3
Marking Holidays

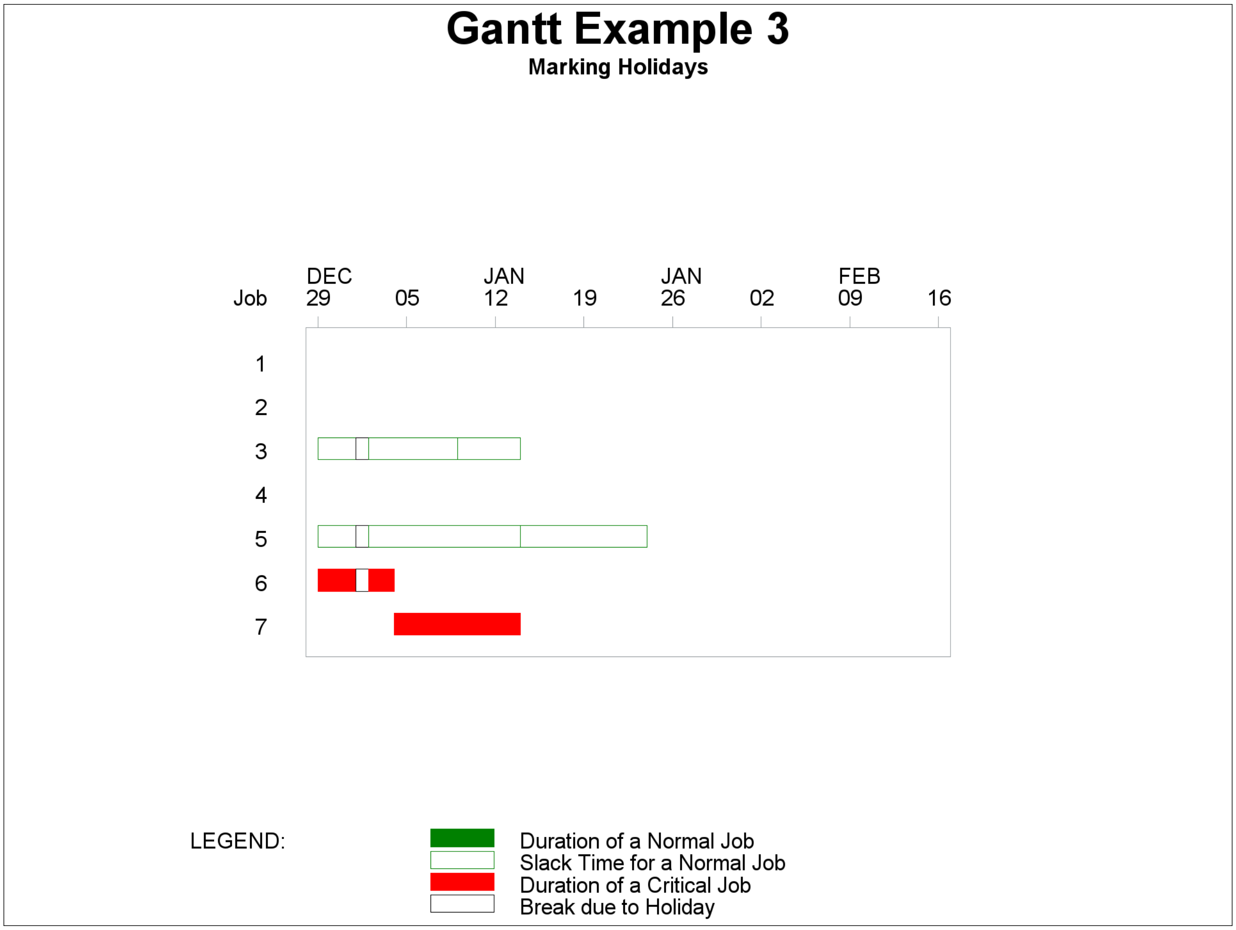
Obs	task	days	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	Approve Plan	5	01DEC03	05DEC03	01DEC03	05DEC03	0	0
2	Drawings	10	06DEC03	15DEC03	06DEC03	15DEC03	0	0
3	Study Market	5	06DEC03	10DEC03	09JAN04	13JAN04	30	0
4	Write Specs	5	06DEC03	10DEC03	11DEC03	15DEC03	5	5
5	Mkt. Strat.	10	11DEC03	20DEC03	14JAN04	23JAN04	30	30
6	Prototype	15	16DEC03	03JAN04	16DEC03	03JAN04	0	0
7	Materials	10	04JAN04	13JAN04	04JAN04	13JAN04	0	0
8	Facility	10	04JAN04	13JAN04	04JAN04	13JAN04	0	0
9	Init. Prod.	10	14JAN04	23JAN04	14JAN04	23JAN04	0	0
10	Evaluate	10	24JAN04	02FEB04	29JAN04	07FEB04	5	5
11	Test Market	15	24JAN04	07FEB04	24JAN04	07FEB04	0	0
12	Marketing	0	24JAN04	24JAN04	13FEB04	13FEB04	20	20
13	Changes	5	08FEB04	12FEB04	08FEB04	12FEB04	0	0
14	Production	0	13FEB04	13FEB04	13FEB04	13FEB04	0	0

```
* plot the schedule;
proc gantt holidata=holidays data=saveh;
   chart / holiday=(holiday) holiend=(holifin);
   id task;
   run;
```

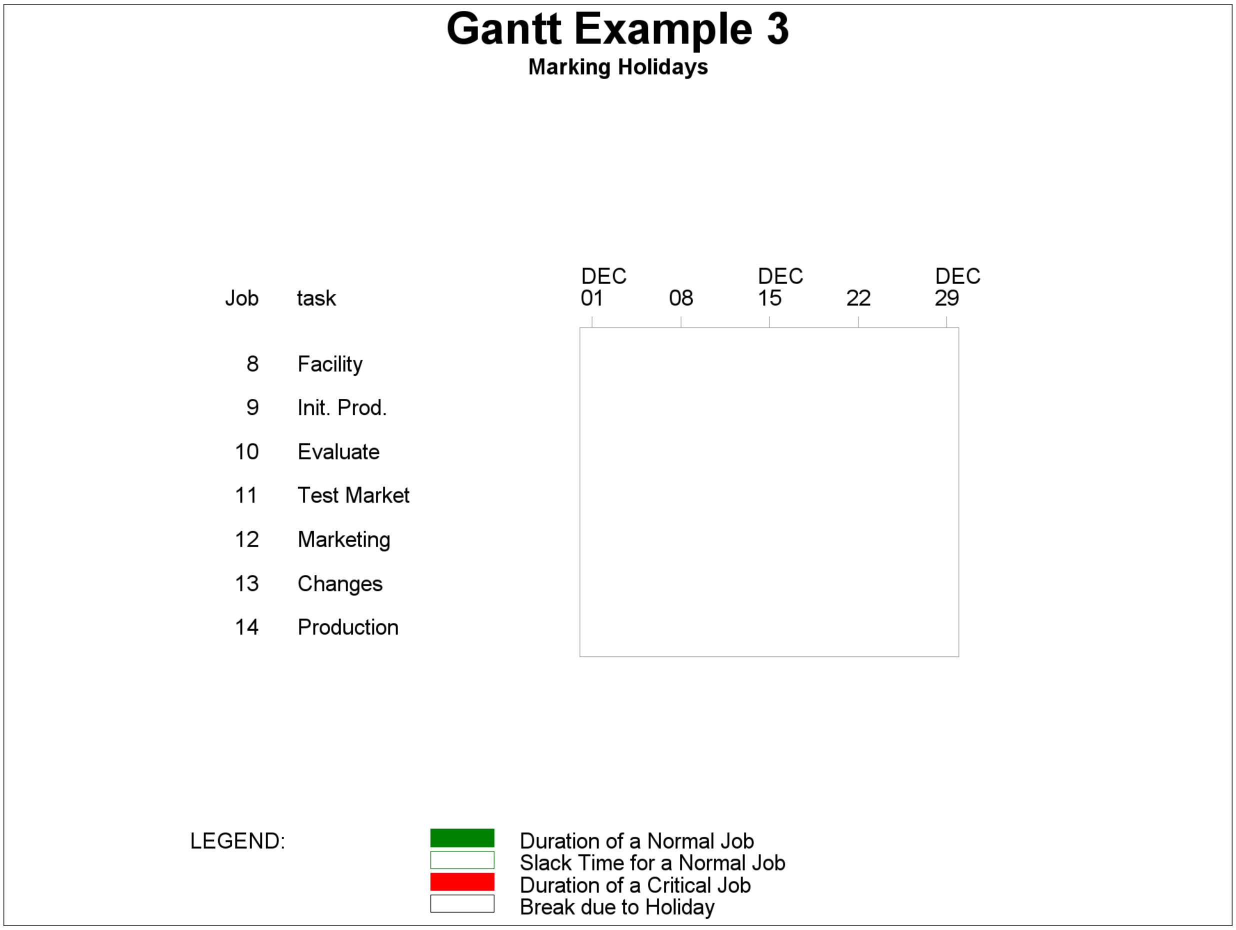
Output 8.3.2 Marking Holidays on the Gantt Chart



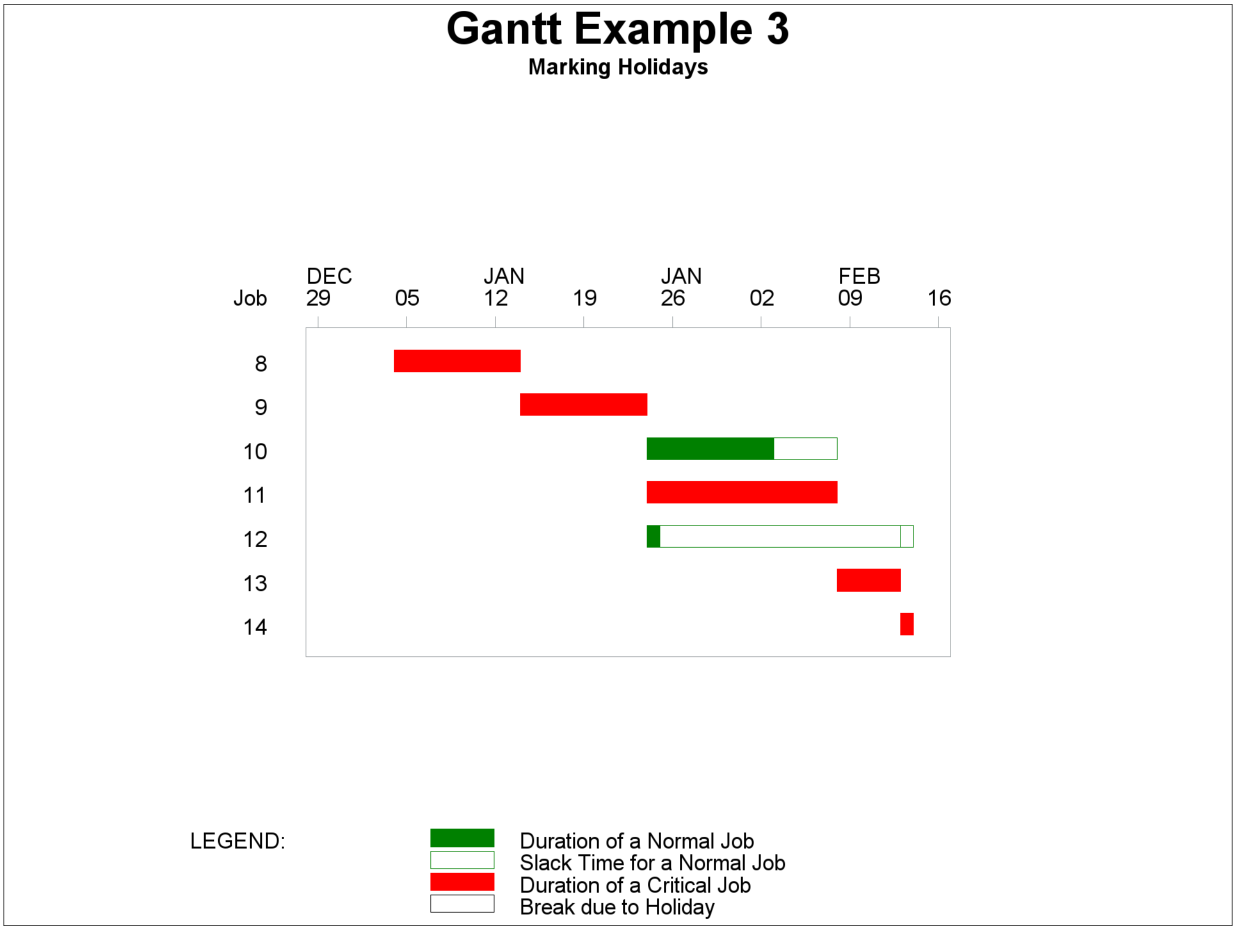
Output 8.3.2 *continued*



Output 8.3.2 *continued*



Output 8.3.2 *continued*



Example 8.4: Marking Milestones and Special Dates

The widget manufacturing project described in Example 8.3 has two activities with zero duration, namely 'Production' and 'Marketing.' By default, PROC GANTT pads finish times by a padding unit, thus these two activities are represented on the Gantt chart as having a duration equal to one day (see the section "Specifying the PADDING= Option" on page 529 for further information on padding). In other words, based on start and finish times alone, PROC GANTT cannot distinguish between activities that are one day or zero days long; it needs knowledge of the activity duration variable, which is specified using the DUR= option in the CHART statement, in order to represent zero duration activities by a milestone symbol.

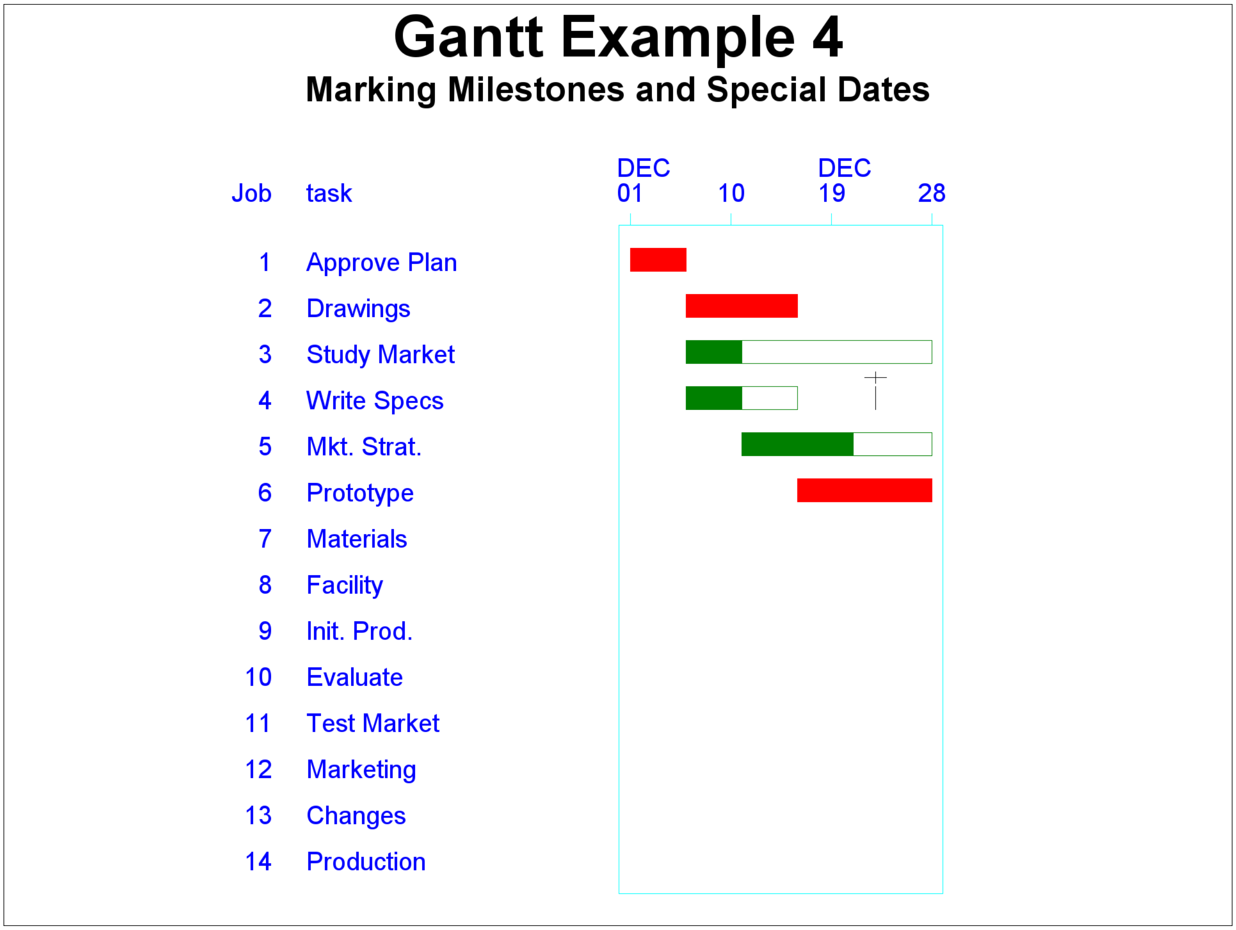
Now, suppose that the Engineering department would like to finish writing up the specifications before Christmas and have the prototype ready by mid-January. In addition, the Marketing department would like to develop a marketing concept by the year's end. The data set, TARGET, contains the target dates for these activities. This data set is merged with the WIDGET data set to produce the WIDGETT data set. The WIDGETT data set is then input to the CPM procedure, which is invoked with an ID statement to ensure that the variable TARGET is passed to the Schedule data set. After sorting the Schedule data set by the early start time, PROC GANTT is used to produce a Gantt chart of the resulting schedule. The Gantt chart is shown in Output 8.4.1.

Before invoking PROC GANTT, you specify the required symbol using a SYMBOL statement. Specifying the variable TARGET in the CHART statement causes target dates to be marked on the chart with the symbol specified in the SYMBOL statement, a PLUS symbol in black. Specifying the DUR= option in the CHART statement causes zero duration schedules to be represented on the chart by the default milestone symbol, a filled diamond. To use a different milestone symbol, use the FMILE= and VMILE= options in the CHART statement. The duration and slack time of the activities are indicated by the use of the appropriate fill patterns as explained in the legend.

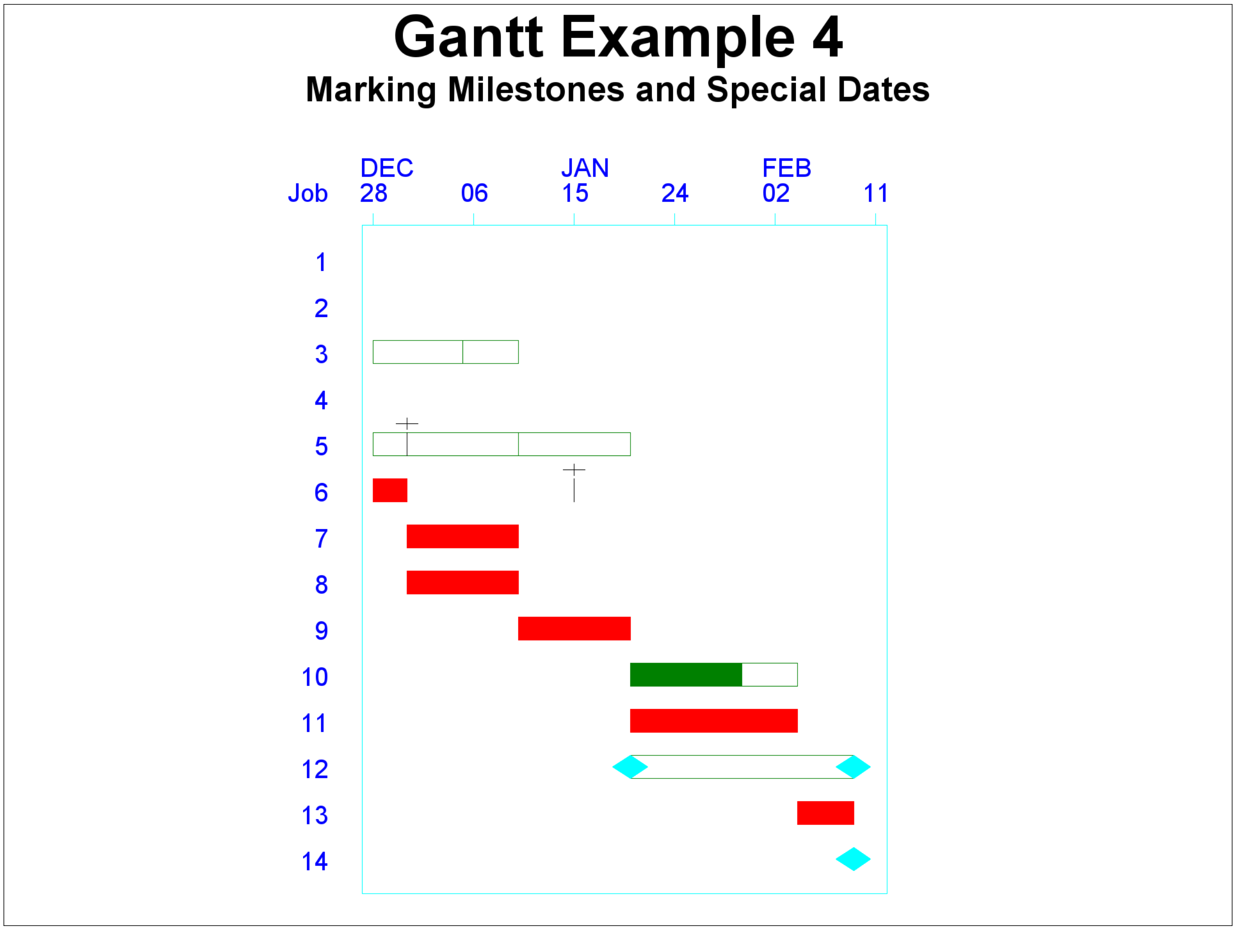
Colors for the milestone, axis, and text are specified using the options CMILE=, CAXIS=, and CTEXT=, respectively.

```
options ps=60 ls=100;

title h=2.5 'Gantt Example 4';
title2 h=1.5 'Marking Milestones and Special Dates';

proc cpm data=widgett date='1dec03'd;
   activity task;
   successor succ1-succ3;
   duration days;
   id target;
   run;

* sort the schedule by the early start date ;
proc sort;
   by e_start;
   run;

goptions htext=1.1 hpos=110 vpos=40;

* set up required pattern and symbol statements;
pattern1 c=green v=s;
pattern2 c=green v=e;
pattern3 c=red   v=s;
symbol   c=black v=plus;

* plot the schedule;
proc gantt;
   chart target / dur=days cmile=cyan
                  ctext=blue caxis=cyan;
   id task;
   run;
```

Output 8.4.1 Marking Milestones and Special Dates in Graphics Mode



Output 8.4.1 *continued*

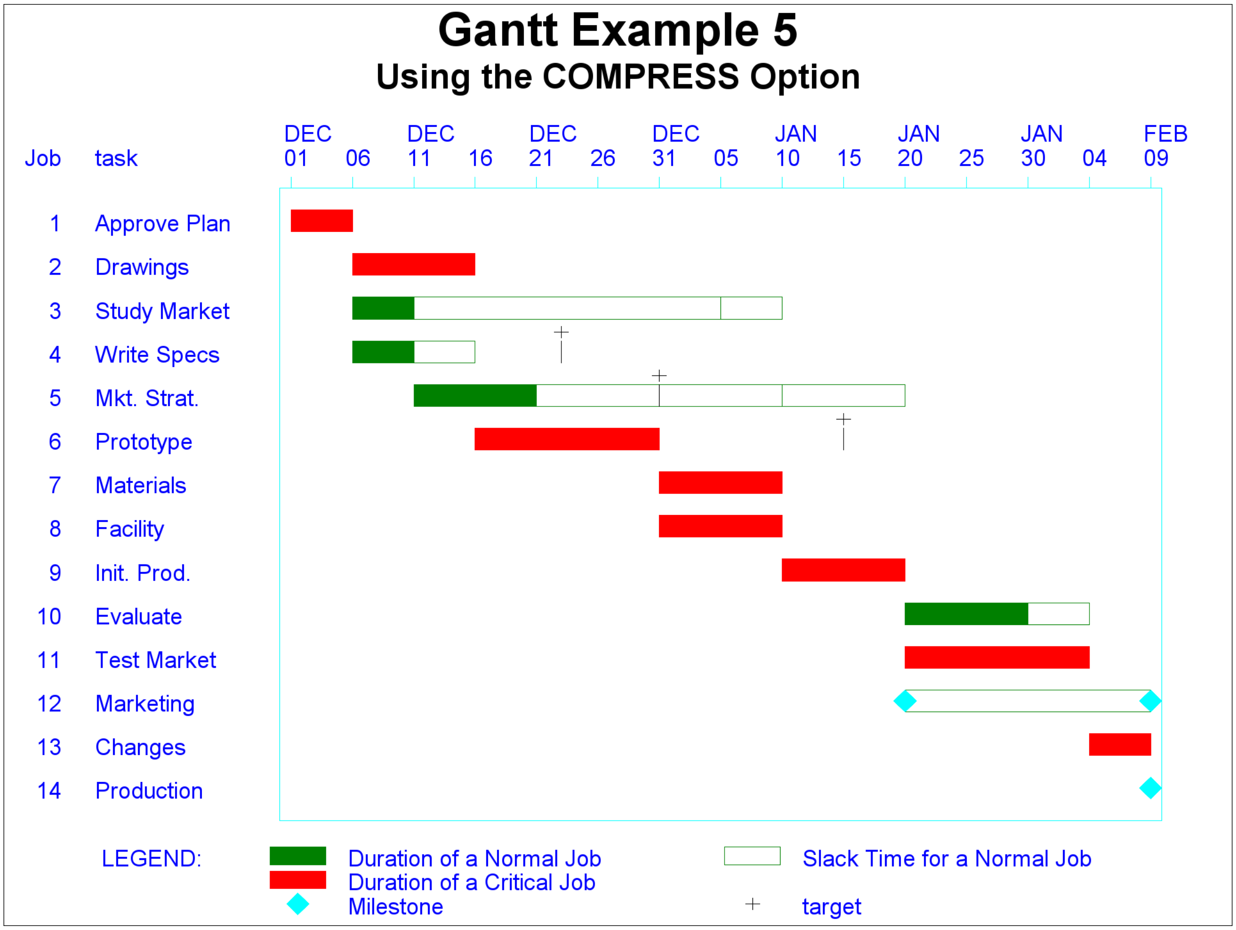


Example 8.5: Using the COMPRESS Option

In the previous example, PROC GANTT produced two pages of output since the chart would not fit on a single page. One way to ensure that the entire chart fits on a single page in graphics mode is to adjust the values of HPOS and VPOS accordingly. An easier way that is independent of the values of HPOS and VPOS is to specify the COMPRESS option in the CHART statement. Output 8.5.1 shows the result of adding the COMPRESS option to the CHART statement in Example 8.4. The PCOMPRESS option would have a similar effect but would also maintain the aspect ratio. Some other options that can be used to control the number of pages generated are the HPAGES= and VPAGES= options.

```
title h=2 'Gantt Example 5';
title2 h=1.5 'Using the COMPRESS Option';

* plot the schedule on one page;
proc gantt;
   chart target / dur=days cmile=cyan
                  ctext=blue caxis=cyan
                  compress;
   id task;
   run;
```

Output 8.5.1 Using the COMPRESS Option



Example 8.6: Using the MININTERVAL= and SCALE= Options

The data sets used for this example are the same as those used to illustrate PROC CPM in Example 4.2. The data set WIDGAOA defines the project using the AOA specification. The data set DETAILS specifies the abbreviated and detailed names for each of the activities in addition to the name of the department that is responsible for that activity. Notice that a dummy activity has been added to the project in order to maintain the precedence relationships established by the WIDGET data set of the previous two examples that define the same project in AON format. The two data sets WIDGAOA and DETAILS are merged to form the WIDGETA data set that is input as the Activity data set to PROC CPM. The data set SAVE produced by PROC CPM and sorted by E_START is shown in Output 8.6.1.

Because MININTERVAL=WEEK and SCALE=10, PROC GANTT uses (1000/*h*)% of the screen width to denote one week, where *h* is the value of HPOS. Note that this choice also causes the chart to become too wide to fit on one page. Thus, PROC GANTT splits the chart into two pages. The first page contains the ID variable as well as the job number while the second page contains only the job number. The chart is split so that the displayed area on each page is approximately equal.

The milestone color is changed to green using the CMILE= option. The resulting Gantt chart is shown in Output 8.6.2.

```
options ps=60 ls=80;

title h=2 'Gantt Example 6';
title2 h=1.5 'Using the MININTERVAL= and SCALE= Options';

data widgaoa;
   format task $12.;
   input task & days tail head;
   datalines;
Approve Plan   5   1   2
Drawings      10   2   3
Study Market   5   2   4
Write Specs    5   2   3
Prototype     15   3   5
Mkt. Strat.   10   4   6
Materials     10   5   7
Facility      10   5   7
Init. Prod.   10   7   8
Evaluate      10   8   9
Test Market   15   6   9
Changes        5   9  10
Production     0  10  11
Marketing      0   6  12
Dummy          0   8   6
;

data details;
   format task $12. dept $13. descrpt $30.;
   input task & dept & descrpt & ;
   label dept = "Department"
         descrpt = "Activity Description";
   datalines;
Approve Plan  Planning       Finalize and Approve Plan
Drawings      Engineering    Prepare Drawings
Study Market  Marketing      Analyze Potential Markets
Write Specs   Engineering    Write Specifications
Prototype     Engineering    Build Prototype
Mkt. Strat.   Marketing      Develop Marketing Concept
Materials     Manufacturing  Procure Raw Materials
Facility      Manufacturing  Prepare Manufacturing Facility
Init. Prod.   Manufacturing  Initial Production Run
Evaluate      Testing        Evaluate Product In-House
Test Market   Testing        Mail Product to Sample Market
Changes       Engineering    Engineering Changes
Production    Manufacturing  Begin Full Scale Production
Marketing     Marketing      Begin Full Scale Marketing
Dummy         .              Production Milestone
;

data widgeta;
   merge widgaoa details;
   run;
```

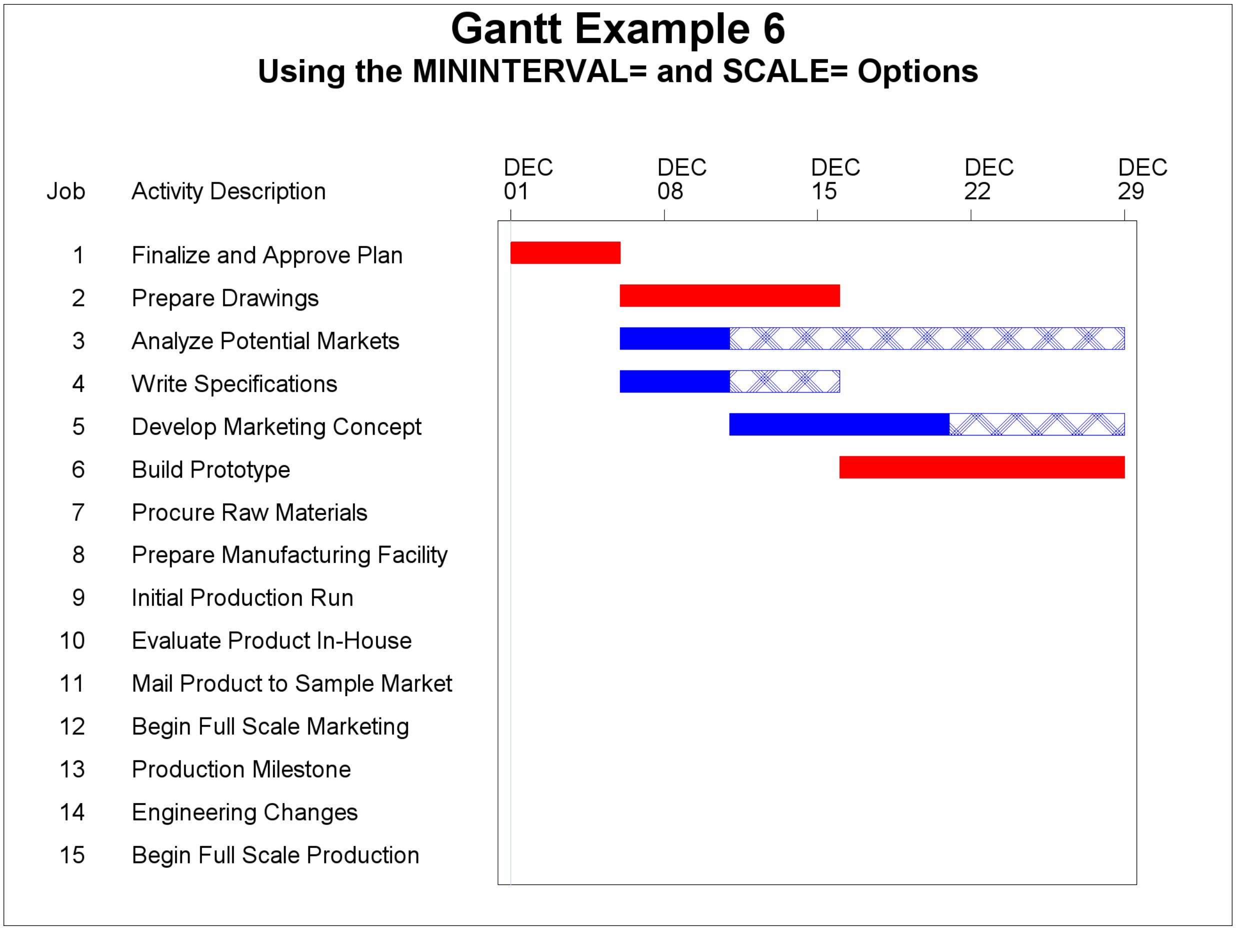
```
* schedule the project;
proc cpm data=widgeta date='1dec03'd out=save;
   tailnode tail;
   headnode head;
   duration days;
   id task dept descrpt;
   run;

* sort the schedule by the early start date ;
proc sort;
   by e_start;
   run;

goptions vpos=43 hpos=80;

* plot the schedule;
proc gantt graphics;
   chart / mininterval=week scale=10 dur=days
           cmile=green nolegend caxis=black
           ref='1dec03'd to '1feb04'd by month;
   id descrpt;
   run;
```

Output 8.6.1 Schedule Data Set SAVE

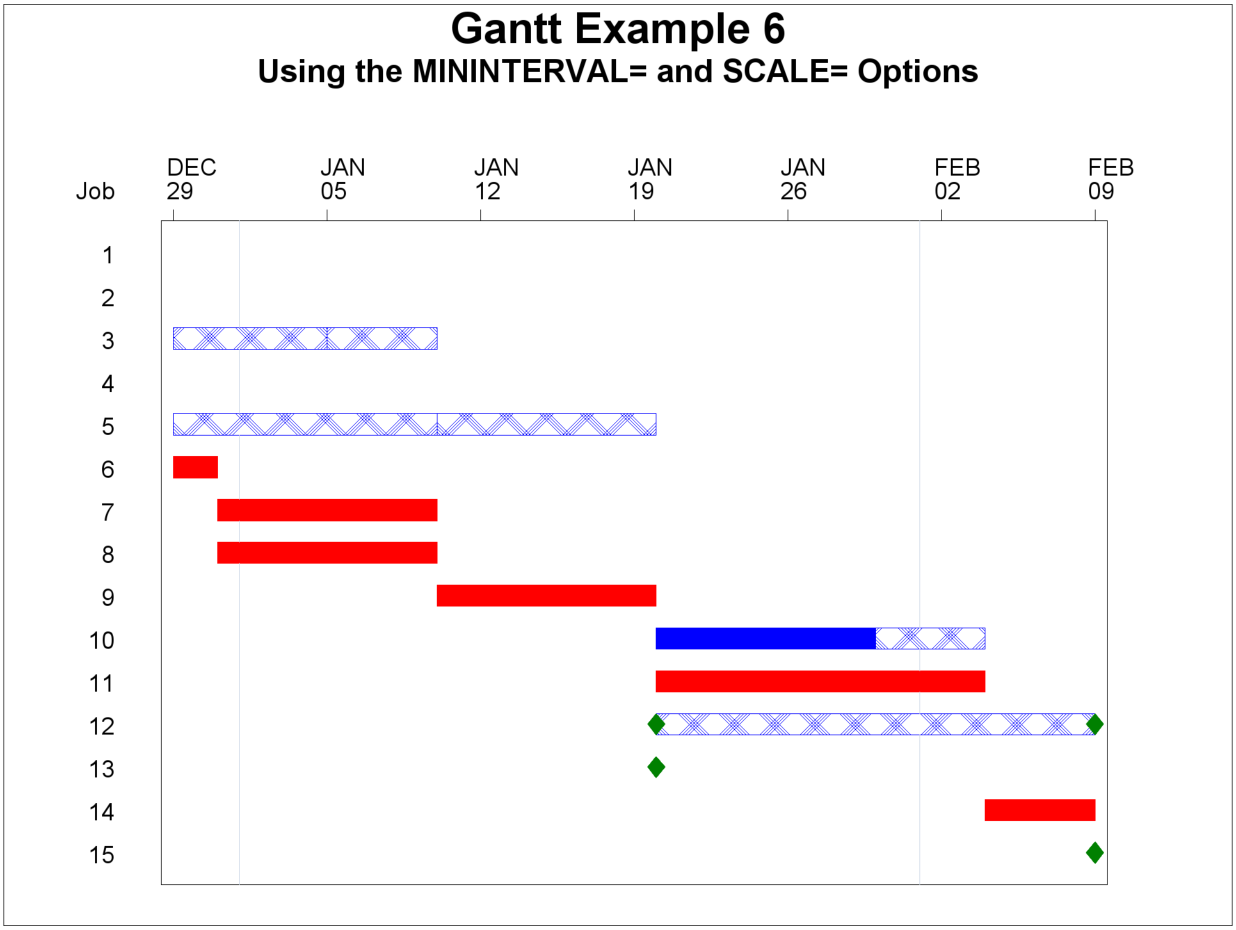
Gantt Example 6
Using the MININTERVAL= and SCALE= Options

descrpt	dept	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Finalize and Approve Plan	Planning	01DEC03	05DEC03	01DEC03	05DEC03	0	0
Prepare Drawings	Engineering	06DEC03	15DEC03	06DEC03	15DEC03	0	0
Analyze Potential Markets	Marketing	06DEC03	10DEC03	05JAN04	09JAN04	30	0
Write Specifications	Engineering	06DEC03	10DEC03	11DEC03	15DEC03	5	5
Develop Marketing Concept	Marketing	11DEC03	20DEC03	10JAN04	19JAN04	30	30
Build Prototype	Engineering	16DEC03	30DEC03	16DEC03	30DEC03	0	0
Procure Raw Materials	Manufacturing	31DEC03	09JAN04	31DEC03	09JAN04	0	0
Prepare Manufacturing Facility	Manufacturing	31DEC03	09JAN04	31DEC03	09JAN04	0	0
Initial Production Run	Manufacturing	10JAN04	19JAN04	10JAN04	19JAN04	0	0
Evaluate Product In-House	Testing	20JAN04	29JAN04	25JAN04	03FEB04	5	5
Mail Product to Sample Market	Testing	20JAN04	03FEB04	20JAN04	03FEB04	0	0
Begin Full Scale Marketing	Marketing	20JAN04	20JAN04	09FEB04	09FEB04	20	20
Production Milestone		20JAN04	20JAN04	20JAN04	20JAN04	0	0
Engineering Changes	Engineering	04FEB04	08FEB04	04FEB04	08FEB04	0	0
Begin Full Scale Production	Manufacturing	09FEB04	09FEB04	09FEB04	09FEB04	0	0

Output 8.6.2 Using the MININTERVAL= and SCALE= Options in Graphics Mode



Output 8.6.2 *continued*

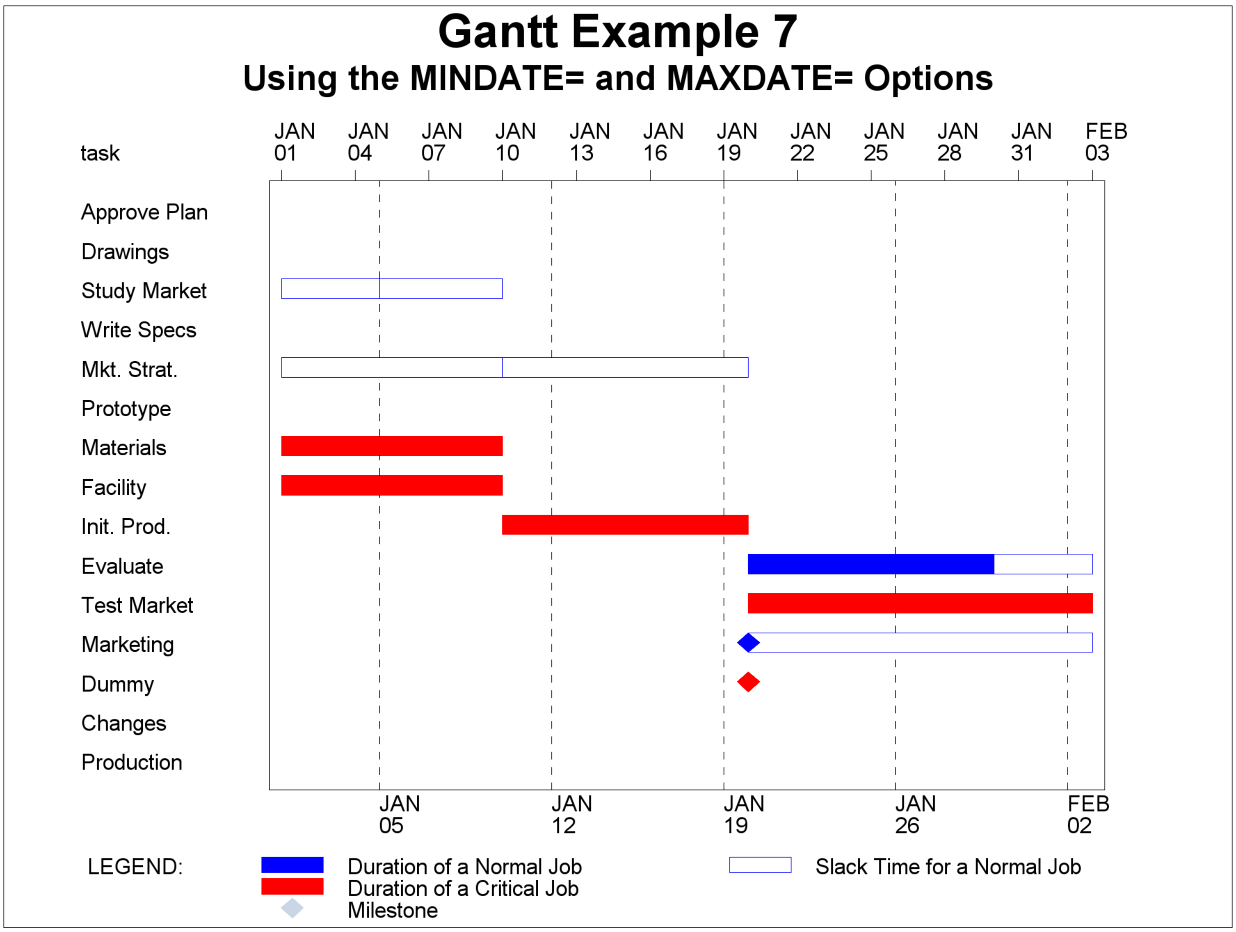


Example 8.7: Using the MINDATE= and MAXDATE= Options

In this example, the SAVE data set from Example 8.6 is used to display the schedule of the project over a limited time period. The start date and end date are specified by the MINDATE= and MAXDATE= options, respectively, in the CHART statement. As in Example 8.5, the COMPRESS option is used to ensure that the region of the Gantt chart lying between January 1, 2004, and February 2, 2004, fits on a single page. The specification REF=‘5JAN04’D TO ‘2FEB04’D BY WEEK causes PROC GANTT to draw reference lines at the start of every week. Further, the reference lines are labeled using the REFLABEL option. The CREF= and LREF= options are specified in the CHART statement to indicate the color and line style, respectively, of the reference lines. The resulting Gantt chart is shown in Output 8.7.1.

```
title h=2 'Gantt Example 7';
title2 h=1.5 'Using the MINDATE= and MAXDATE= Options';

goptions vpos=40 hpos=100;

* plot the schedule;
proc gantt graphics data=save;
   chart / mindate='1jan04'd maxdate='2feb04'd
           ref='5jan04'd to '2feb04'd by week
           reflabel cref=black lref=2 caxis=black
           compress dur=days nojobnum;
   id task;
   run;
```

Output 8.7.1 Using the MINDATE= and MAXDATE= Options in Graphics Mode

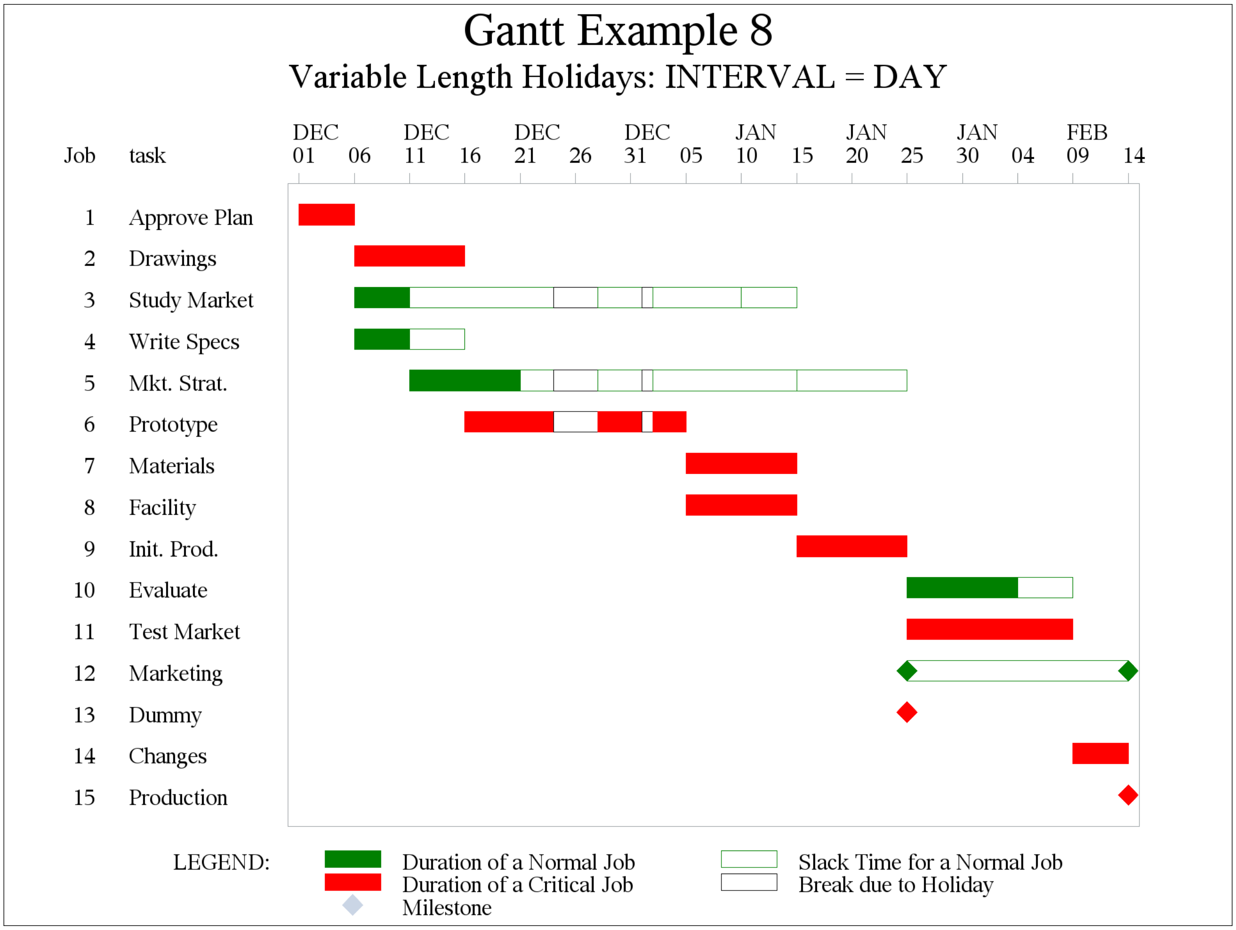


Example 8.8: Variable-Length Holidays

This example shows how you can mark vacation periods that last longer than one day on the Gantt chart. This can be done by using the HOLIDUR= option in the CHART statement. Recall that holiday duration is assumed to be in *interval* units where *interval* is the value specified for the INTERVAL= option. The project data for this example are the same as the data used in the previous example. Suppose that in your scheduling plans you want to assign work on all days of the week, allowing for a Christmas vacation of four days starting from December 24, 2003, and a day off on January 1, 2004 for the New Year. The data set HOLIDAYS contains the holiday information for the project. First, the project is scheduled with INTERVAL=DAY so that the holidays are on December 24, 25, 26, and 27, 2003, and on January 1, 2004. PROC GANTT is invoked with INTERVAL=DAY to correspond to the invocation of PROC CPM. The desired font is specified by using the FONT= option in the CHART statement and the F= option in the TITLE statement. As an alternative, the desired font can be specified globally by using the FTEXT= option in a GOPTIONS statement. The resulting Gantt chart is shown in Output 8.8.1.

```
data holidays;
   format holiday holifin date7.;
   input holiday & date7. holifin & date7. holidur;
   datalines;
24dec03  27dec03  4
01jan04  .        .
;

 * schedule the project subject to holidays;
proc cpm data=widgeta holidata=holidays out=sched1
         date='1dec03'd interval=day;
   tailnode tail;
   headnode head;
   duration days;
   id task dept descrpt;
   holiday  holiday / holidur=(holidur);
   run;

 * sort the schedule by the early start date ;
proc sort;
   by e_start;
   run;

 * plot the schedule;
title h=2 f='Thorndale AMT' 'Gantt Example 8';
title2 h=1.5 f='Thorndale AMT' 'Variable Length Holidays: INTERVAL = DAY';
proc gantt holidata=holidays data=sched1 ;
   chart / holiday=(holiday) holidur=(holidur) font='Thorndale AMT'
           dur=days interval=day pcompress;
   id task;
   run;
```

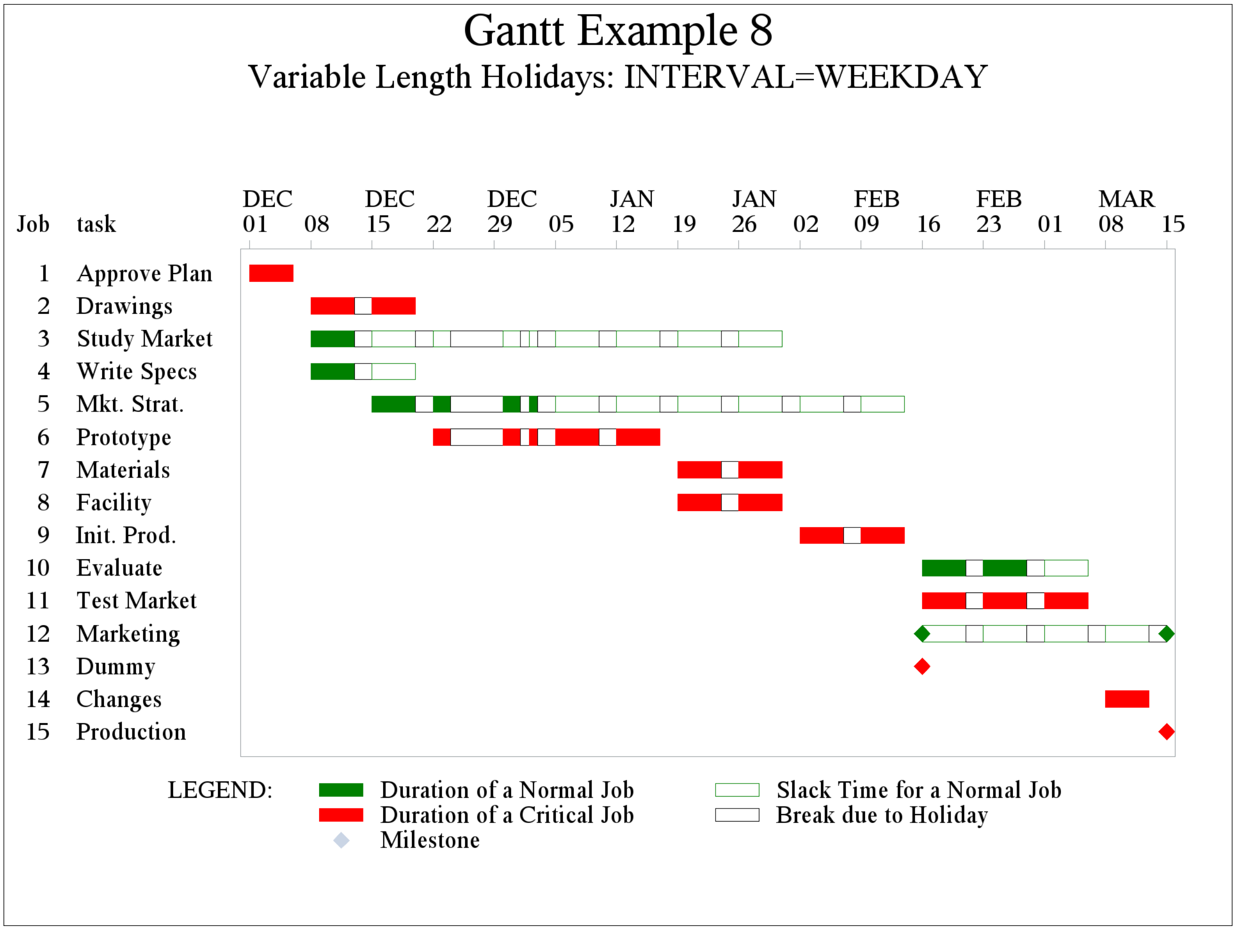
Output 8.8.1 Variable Length Holidays: INTERVAL=DAY



Next, consider the same project and Holiday data set, but invoke PROC CPM with INTERVAL=WEEKDAY. Then, the value '4' specified for the variable HOLIDUR is interpreted as 4 weekdays. The holidays are on December 24, 25, 26, and 29, 2003, and on January 1, 2004, because December 27 and 28 (Saturday and Sunday) are non-working days. The same steps are used as previously, except that INTERVAL is set to WEEKDAY instead of DAY in both PROC CPM and PROC GANTT. Suppose that the resulting data set is saved as SCHED2. The following invocation of PROC GANTT produces Output 8.8.2. Note that the use of INTERVAL=WEEKDAY causes weekends to be also marked on the chart.

```
title2 h=1.5 f='Thorndale AMT' 'Variable Length Holidays: INTERVAL=WEEKDAY';

proc gantt holidata=holidays data=sched2;
   chart / holiday=(holiday) holidur=(holidur)
           font='Thorndale AMT'
           height=1.4
           interval=weekday
           dur=days
           pcompress;
   id task;
   run;
```

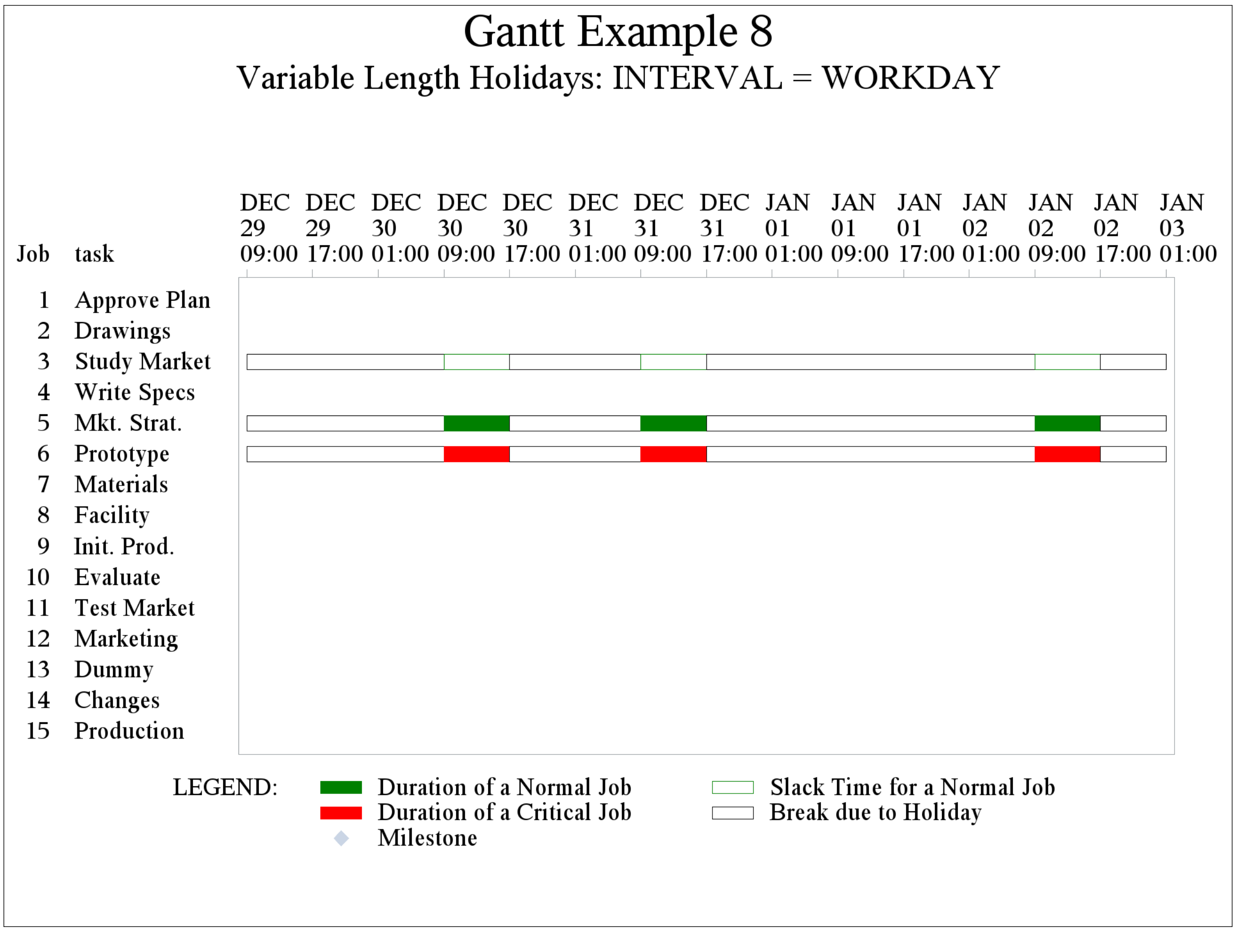
Output 8.8.2 Variable Length Holidays: INTERVAL=WEEKDAY



Finally, when the INTERVAL= option is specified as WORKDAY, the workday is assumed to be from 9:00 a.m. to 5:00 p.m., and the Christmas holiday period begins at 5:00 p.m. on December 23, 2003, and ends at 9:00 a.m. on December 30, 2004. PROC GANTT is invoked with the MARKBREAK option and MININTERVAL=DTHOUR so that all breaks during a day can be seen. Because the SCALE= option is not specified, each column denotes one hour of the schedule. Since the project duration is several days long, the entire Gantt chart would be spread across many pages. Simply specifying the COMPRESS or PCOMPRESS option will not be of much help since the text would be barely legible owing to the extent of the scaling. Hence, only a portion of the Gantt chart is shown in Output 8.8.3 using the MINDATE= and MAXDATE= options. Note that the Gantt chart is labeled with the date as well as the time values on the time axis.

```
title2 h=1.5 f='Thorndale AMT' 'Variable Length Holidays: INTERVAL = WORKDAY';

proc gantt holidata=holidays data=sched3;
   chart / holiday=(holiday) holidur=(holidur)
           dur=days interval=workday
           font='Thorndale AMT'
           mininterval=dthour markbreak
           mindate='29dec03:09:00:00'dt
           maxdate='03jan04:00:00:00'dt
           pcompress height=1.5;
   id task;
   run;
```

Output 8.8.3 Variable Length Holidays: INTERVAL=WORKDAY



Example 8.9: Multiple Calendars

This example illustrates the use of multiple calendars within a project. The data for this example are the same as the data used in Example 4.10 to illustrate the CPM Procedure. The input data sets to PROC CPM are displayed in Output 8.9.1. The WORKDATA data set defines several shift patterns, which in turn are identified with four different calendars in the CALEDATA data set:

- The 'DEFAULT' calendar has five 8-hour workdays (8 a.m. - 4 p.m.) on Monday through Friday and holidays on Saturday and Sunday.
- The 'OVT_CAL' calendar defines the "overtime" calendar that is followed by the Engineering department to build the prototype. The 'OVT_CAL' calendar has five 10-hour workdays (8 a.m. - 6 p.m.) on Monday through Friday, a 4-hour halfday (8 a.m. - 12 noon) on Saturday and a holiday on Sunday.
- The 'PROD_CAL' calendar defines the "production" calendar that is used for full-scale production of the widget. The 'PROD_CAL' calendar consists of continuous work from Monday 8 a.m. through Saturday 6 p.m. except for two 2-hour breaks per day from 6 a.m. to 8 a.m. and from 6 p.m. to 8 p.m. Thus, 'PROD_CAL' is made up of eleven 8-hour shifts per week; six day shifts and five night shifts.
- The 'Eng_cal' calendar defines the calendar followed by the Engineering department for writing the specifications for the prototype. The 'Eng_cal' calendar has the same work pattern as the default calendar with an extra holiday period of seven days starting on December 8, 2003.

The HOLIDATA data set defines the appropriate holidays for the different calendars. The project data set WIDGVAC includes a variable named CAL to identify the appropriate calendar for each activity.

Output 8.9.1 Multiple Calendars: Data Sets

Multiple Calendars

Workdays Data Set

Obs	fullday	halfday	ovtday	s1	s2	s3
1	8:00	8:00	8:00	.	8:00	.
2	16:00	12:00	18:00	6:00	18:00	6:00
3	.	.	.	8:00	20:00	8:00
4	.	.	.	18:00	.	18:00
5	.	.	.	20:00	.	.
6	.	.	.	.	.	.

Calendar Data Set

Obs	cal	_sun_	_mon_	_tue_	_wed_	_thu_	_fri_	_sat_
1	DEFAULT	holiday	fullday	fullday	fullday	fullday	fullday	holiday
2	OVT_CAL	holiday	ovtday	ovtday	ovtday	ovtday	ovtday	halfday
3	PROD_CAL	holiday	s2	s1	s1	s1	s1	s3
4	Eng_cal							

Holidays Data Set

Obs	holiday	holifin	holidur	cal
1	08DEC03	.	7	Eng_cal
2	24DEC03	26DEC03	.	
3	01JAN04	01JAN04	.	

Output 8.9.1 *continued*

Project Data Set

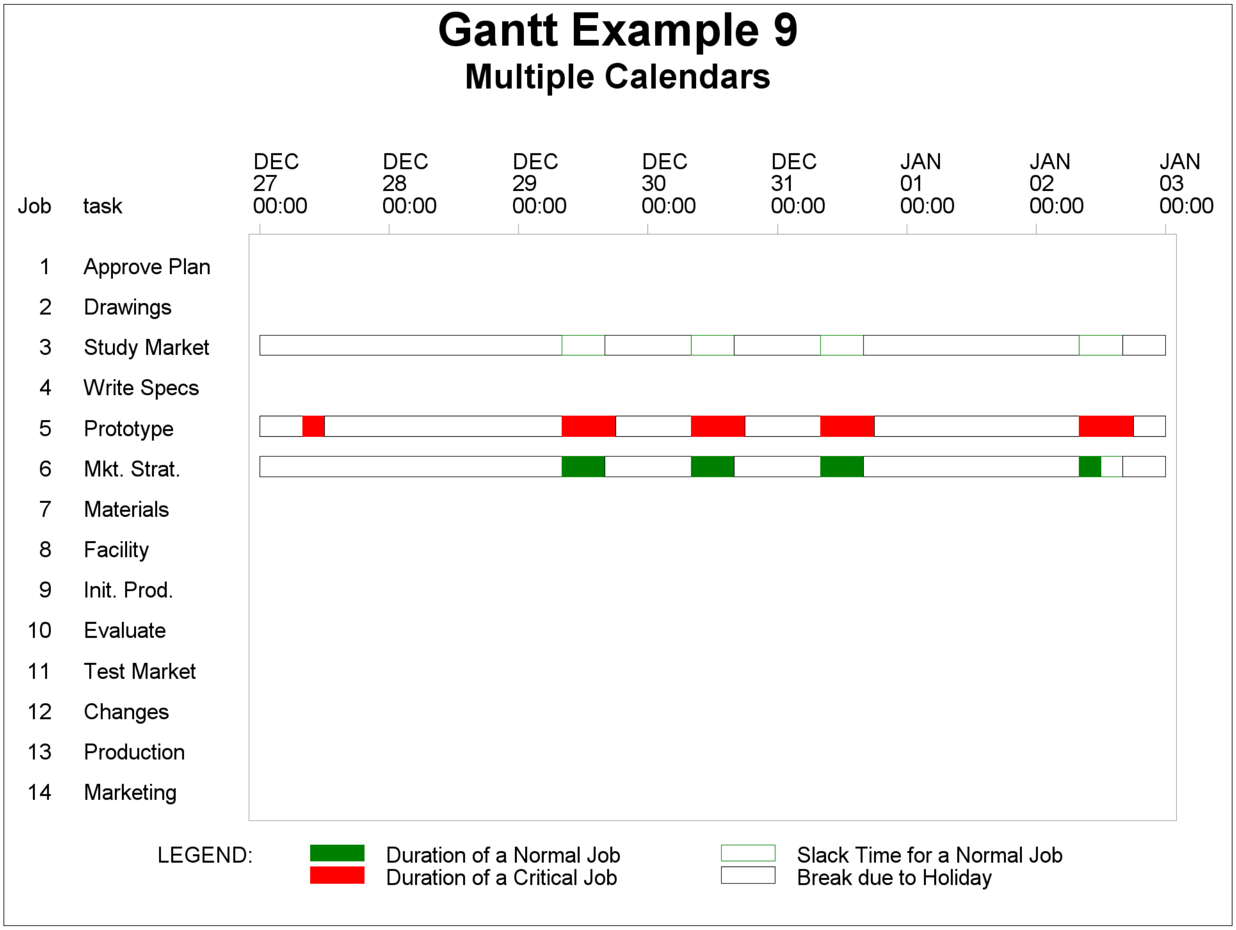
Obs	task	succ1	succ2	succ3	days	cal
1	Approve Plan	Drawings	Study Market	Write Specs	5.5	DEFAULT
2	Drawings	Prototype			10.0	DEFAULT
3	Study Market	Mkt. Strat.			5.0	DEFAULT
4	Write Specs	Prototype			4.5	Eng_cal
5	Prototype	Materials	Facility		15.0	OVT_CAL
6	Mkt. Strat.	Test Market	Marketing		10.0	DEFAULT
7	Materials	Init. Prod.			10.0	DEFAULT
8	Facility	Init. Prod.			10.0	DEFAULT
9	Init. Prod.	Test Market	Marketing	Evaluate	10.0	DEFAULT
10	Evaluate	Changes			10.0	DEFAULT
11	Test Market	Changes			15.0	DEFAULT
12	Changes	Production			5.0	DEFAULT
13	Production				0.0	PROD_CAL
14	Marketing				0.0	DEFAULT

The program used to invoke PROC CPM and PROC GANTT follows. The CALENDAR= and WORKDAY= options are specified in the PROC GANTT statement to identify the CALEDATA and WORKDATA data sets, respectively. The CALID= option in the CHART statement names the variable identifying the calendar that each observation refers to in the WIDGVAC and CALEDATA data sets. Since the value of MININTERVAL= is DTDAY, setting the SCALE= value to 12 ensures that a single column on the Gantt chart represents two hours. This is done in order to be able to detect a two hour difference between schedules. Consequently, the MINDATE= and MAXDATE= options are used to control the output produced by PROC GANTT. The resulting Gantt chart is shown in Output 8.9.2. Notice the 5 column duration for 'Prototype' on December 29, 2003 representing a 10-hour day versus the 4 column duration for 'Mkt. Strat.' for the same day representing 8 hours of work. Although MAXDATE= is set to 8 a.m. on January 2, 2004, the last tick mark is the beginning of January 3, 2004. This is because the specified value of the MAXDATE= option does not correspond to a tick mark (based on the SCALE= and MININTERVAL= options); the value used is the first tick mark appearing after the value of the MAXDATE= option.

```
proc cpm date='01dec03'd interval=workday data=widgvac
         out=schedvac holidata=holidata
         workday=workdata calendar=caledata;
   holiday holiday / holifin=holifin holidur=holidur;
   activity task;
   duration days;
   successor succ1 succ2 succ3;
   calid cal;
   run;

title h=2 'Gantt Example 9';
title2 h=1.5 'Multiple Calendars';

proc gantt data=schedvac holidata=holidata
           workday=workdata calendar=caledata ;
   chart / holiday=(holiday) holiend=(holifin)
           calid=cal
```

```
            markbreak scale=12
            mindate='27dec03:00:00'dt
            maxdate='02jan04:08:00'dt
            pcompress;
   id task;
   run;
```

Output 8.9.2 Multiple Calendars



Example 8.10: Plotting the Actual Schedule

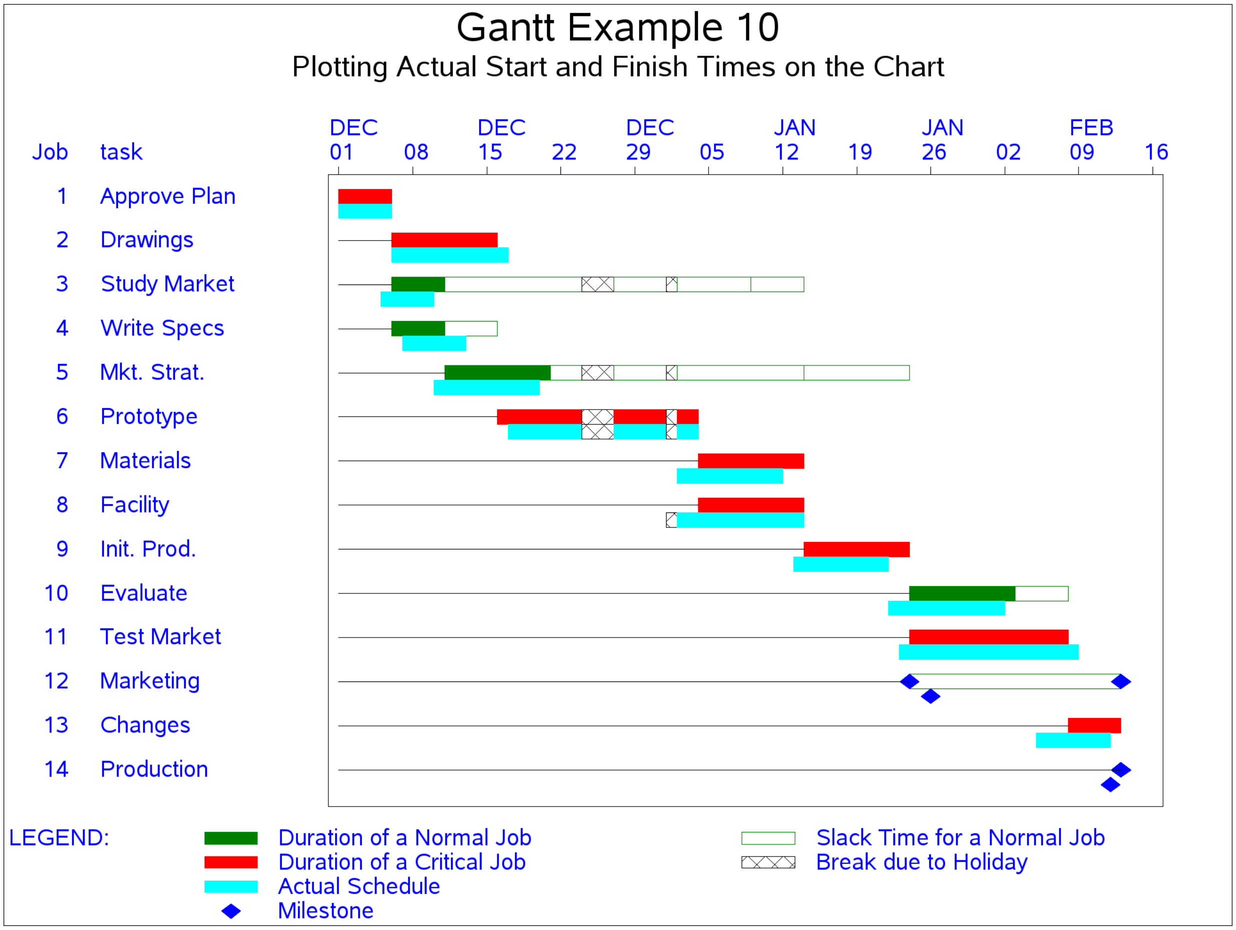
Suppose that the project is complete and you want to compare the actual progress of the activities with the predicted schedule computed by PROC CPM. The following DATA step stores the actual start and finish times of each activity in a data set named COMPLETE. A data set named WIDGELA is then created that contains both the schedule obtained from PROC CPM (the data set SAVEH from Example 8.3 is used because it does not contain the dummy activity) and the actual schedule. The resulting data set is sorted by early start time.

Fill patterns are specified using PATTERN statements, and the COMPRESS option is employed in order to draw the entire Gantt chart on one page. Predicted schedules as well as actual schedules are plotted on separate bars for each activity. The A_START= and A_FINISH= options in the CHART statement are used to specify the variables containing the actual start and finish times for each activity. The actual schedule is plotted with the fill pattern specified in the sixth PATTERN statement. This example also illustrates the drawing of holidays in graphics mode. PROC GANTT uses the fill pattern specified in the seventh PATTERN statement to represent the holidays defined by the HOLIDATA= data set. The holidays are identified to PROC GANTT by specifying the HOLIDAY= and HOLIFIN= options in the CHART statement.

The HCONNECT option causes a connecting line to be drawn from the left boundary of the chart to the early start time for each activity. The CHCON= option specifies the color for drawing the connect lines. You can use the LHCON= option in the CHART statement to specify a line style other than the default style for the connect lines. The Gantt chart is shown in Output 8.10.1.

```
data complete;
   format activity $12. sdate date7. fdate date7.;
   input activity & sdate & date7. fdate & date7.;
   datalines;
Approve Plan   01dec03  05dec03
Drawings       06dec03  16dec03
Study Market   05dec03  09dec03
Write Specs    07dec03  12dec03
Prototype      17dec03  03jan04
Mkt. Strat.    10dec03  19dec03
Materials      02jan04  11jan04
Facility       01jan04  13jan04
Init. Prod.    13jan04  21jan04
Evaluate       22jan04  01feb04
Test Market    23jan04  08feb04
Changes        05feb04  11feb04
Production     12feb04  12feb04
Marketing      26jan04  26jan04
;

 * merge the computed schedule with the actual schedule;
data widgela;
   merge saveh complete;

* sort the data;
proc sort;
   by e_start;
   run;

* set vpos to 40 and hpos to 100;
goptions vpos=40 hpos=100;

title h=1.75 f='Albany AMT' 'Gantt Example 10';
title2 h=1.25
   f='Albany AMT' 'Plotting Actual Start and Finish Times on the Chart';

* plot the computed and actual schedules using proc gantt;
proc gantt graphics data=widgela holidata=holidays;
   chart / holiday=(holiday) holifin=(holifin)
           a_start=sdate a_finish=fdate
           dur=days font='Albany AMT' chcon=black
           hconnect compress ctext=blue height=1.5
           caxis=black cmile=blue;
   id task;
   run;
```

Output 8.10.1 Plotting the Actual Schedule on the Gantt Chart



Example 8.11: Comparing Progress Against a Baseline Schedule

Suppose that the widget manufacturing project is currently in progress and you want to measure its performance by comparing it with a baseline schedule. For example, the baseline schedule may be the originally planned schedule, a target schedule that you would like to achieve, or an existing schedule that you intend to improve on. The data for this example come from Example 4.13, which was used to illustrate the options available in PROC CPM. Prior to the beginning of the project, the predicted early schedule is saved by PROC CPM as the baseline schedule. Progress information for the project as of December 19, 2003, is saved in the ACTUAL data set. The variables SDATE and FDATE represent the actual start and actual finish times, respectively. The variables PCTC and RDUR represent the percent of work completed and the remaining days of work for each activity, respectively. PROC CPM is then invoked using the baseline and project progress information with TIMENOW set to December 19, 2003. The scheduling is carried out with the AUTOPUPDT option in order to automatically update progress information. The Schedule data set WIDGUPDT produced by PROC CPM is shown in Output 8.11.1. Notice that the development of a marketing strategy (activity 5: ‘Mkt. Strategy’) and the building of the prototype (activity 6: ‘Prototype’) have a specified value for A_START and a missing value for A_FINISH, indicating that they are currently in progress at TIMENOW.

PROC GANTT is next invoked with the data set WIDGUPDT. This data set contains the actual schedule variables A_START and A_FINISH and the baseline schedule variables B_START and B_FINISH. The Gantt chart is drawn with three schedule bars per activity. The first bar represents the predicted early/late schedule based on the actual data specified, the second bar represents the actual schedule, and the third bar represents the baseline schedule. The TIMENOW= option is specified in the CHART statement to draw a timenow line on December 19, 2003. Actual schedule bars for 'Mkt. Strategy' and 'Prototype' are drawn up to TIMENOW to indicate that they are currently in progress. You can use the CTNOW=, LTNOW=, and WTNOW= options to change the color, style, and width of the timenow line, respectively. To suppress the timenow label displayed at the bottom of the axis, specify the NOTNLABEL in the CHART statement.

```
title h=1.2 'Gantt Example 11';

* estimate schedule based on actual data;
proc cpm data=widgact holidata=holidays
         out=widgupdt date='1dec03'd;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   baseline / compare=early;
   actual / as=sdate af=fdate timenow='19dec03'd
            remdur=rdur pctcomp=pctc
            autoupdt;
   run;

* sort the data;
proc sort;
   by e_start;
   run;

* print the data;
title2 'Progress Data';

proc print;
   var task e_: l_: a_start a_finish b_: ;
   run;

title2 'Comparing Project Progress against a Baseline Schedule';

* plot the actual and baseline schedules using proc gantt;
proc gantt data=widgupdt holidata=holidays;
   chart / holiday=(holiday) holifin=(holifin)
           timenow='19dec03'd dur=days
           scale=2 height=1.6
           pcompress;
   id task;
   run;
```

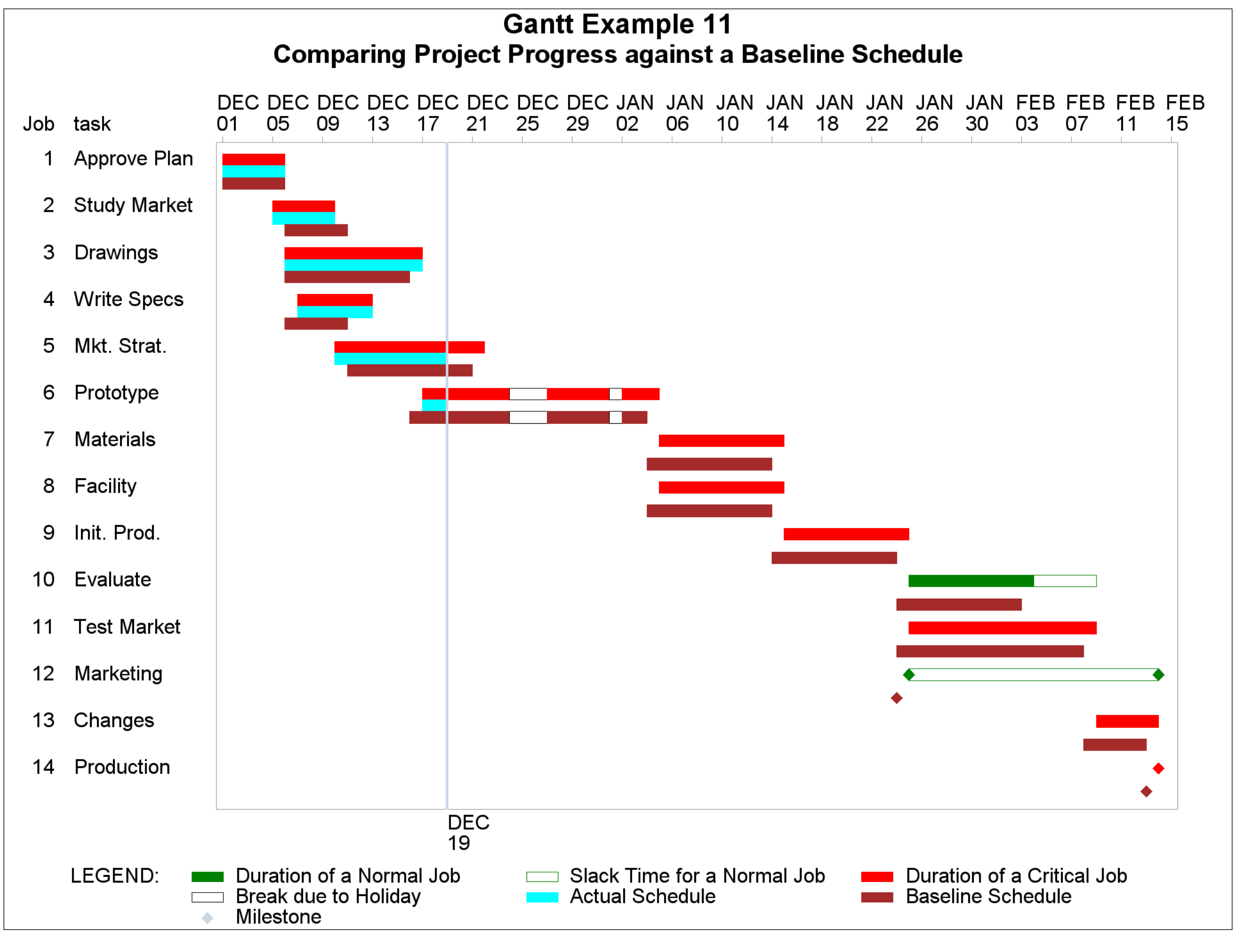
Output 8.11.1 Schedule Data Set WIDGUPDT

Gantt Example 11

Progress Data

Obs	task	E_START	E_FINISH	L_START	L_FINISH	A_START	A_FINISH	B_START	B_FINISH
1	Approve Plan	01DEC03	05DEC03	01DEC03	05DEC03	01DEC03	05DEC03	01DEC03	05DEC03
2	Study Market	05DEC03	09DEC03	05DEC03	09DEC03	05DEC03	09DEC03	06DEC03	10DEC03
3	Drawings	06DEC03	16DEC03	06DEC03	16DEC03	06DEC03	16DEC03	06DEC03	15DEC03
4	Write Specs	07DEC03	12DEC03	07DEC03	12DEC03	07DEC03	12DEC03	06DEC03	10DEC03
5	Mkt. Strat.	10DEC03	21DEC03	10DEC03	21DEC03	10DEC03	.	11DEC03	20DEC03
6	Prototype	17DEC03	04JAN04	17DEC03	04JAN04	17DEC03	.	16DEC03	03JAN04
7	Materials	05JAN04	14JAN04	05JAN04	14JAN04	.	.	04JAN04	13JAN04
8	Facility	05JAN04	14JAN04	05JAN04	14JAN04	.	.	04JAN04	13JAN04
9	Init. Prod.	15JAN04	24JAN04	15JAN04	24JAN04	.	.	14JAN04	23JAN04
10	Evaluate	25JAN04	03FEB04	30JAN04	08FEB04	.	.	24JAN04	02FEB04
11	Test Market	25JAN04	08FEB04	25JAN04	08FEB04	.	.	24JAN04	07FEB04
12	Marketing	25JAN04	25JAN04	14FEB04	14FEB04	.	.	24JAN04	24JAN04
13	Changes	09FEB04	13FEB04	09FEB04	13FEB04	.	.	08FEB04	12FEB04
14	Production	14FEB04	14FEB04	14FEB04	14FEB04	.	.	13FEB04	13FEB04

Output 8.11.2 Comparing Project Progress Against a Baseline Schedule



Example 8.12: Using the COMBINE Option

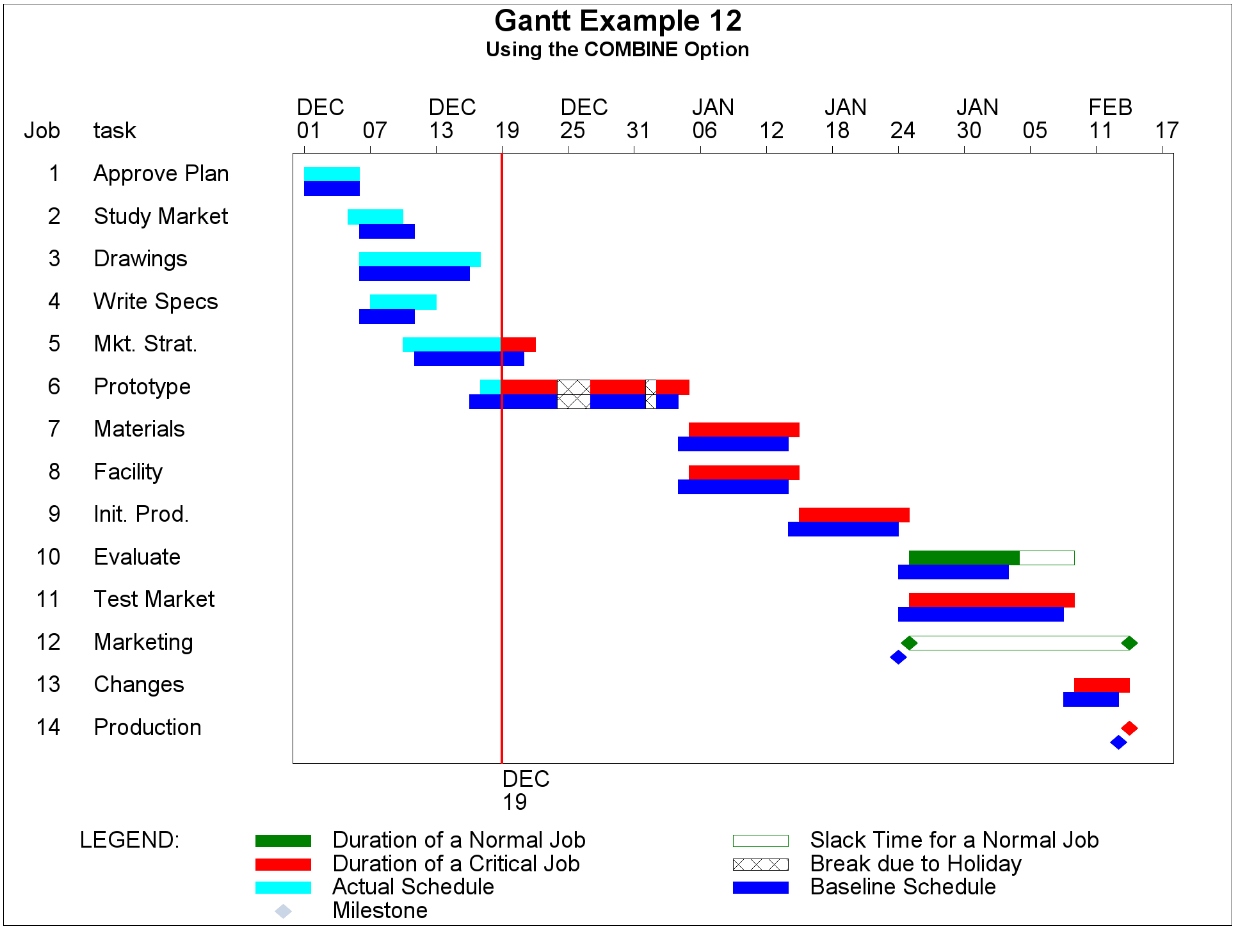
When you monitor a project in progress, as in the previous example, it is evident that there are no actual dates beyond TIMENOW and that PROC CPM sets the early times to the corresponding actual times for activities that are completed or in progress (see Output 8.11.1). For example, activities 1 through 4 have their early schedule equal to the actual schedule. Activities 5 and 6 have their early start equal to the actual start; however the actual finish for these two activities is missing since they are in progress at TIMENOW. Finally, activities 7 through 14 have no actual information.

The COMBINE option in PROC GANTT exploits the fact that the early times are made consistent with the actual times to strip away a lot of the redundancy and produce a more compact Gantt chart while retaining all of the essential schedule information. Specifying the COMBINE option in the CHART statement of the previous example produces the Gantt chart in Output 8.12.1. Instead of using two separate bars to draw the early/late schedule and the actual schedule, the COMBINE option causes PROC GANTT to use one bar to represent all three schedules and draws a timenow line. The actual schedule is shown to the left of TIMENOW and the early/late schedule is shown to the right of TIMENOW. Thus, for activities 1 through 4, the actual schedule is drawn on the first bar to the left of the timenow line. Activities 5 and 6 are in progress at TIMENOW, which is indicated by the actual start positioned to the left of TIMENOW and the predicted early/late schedule, based on the progress made up to TIMENOW, drawn to the right of TIMENOW. Activities 7 through 14 have not yet started, and this is reflected in their predicted early/late schedules drawn to the right of TIMENOW.

The COMBINE option draws a timenow line by default, and if the TIMENOW= option is not specified, the procedure computes the value of TIMENOW based on the schedule data as explained in the "Syntax" section. In this example, specifying the COMBINE option without the TIMENOW= option causes a timenow line to be drawn on December 18, 2003, since this is the first day following the largest actual value. The CTNOW= option is used to specify the color of the timenow line. You can change the line style and line width of the timenow line by specifying the LTNOW= and WTNOW= options, respectively, in the CHART statement.

```
title h=1.6 'Gantt Example 12';

title2 'Using the COMBINE Option';

* set vpos to 50 and hpos to 100;
goptions vpos=50 hpos=100;

* plot the combined and baseline schedules using proc gantt;
proc gantt graphics data=widgupdt holidata=holidays;
   chart / holiday=(holiday) holifin=(holifin)
           compress ctnow=red caxis=black
           height=1.5
           timenow='19dec03'd
           dur=days
           combine;
   id task;
   run;
```

Output 8.12.1 Using the COMBINE Option in Graphics Mode



Example 8.13: Plotting the Resource-Constrained Schedule

This example illustrates plotting the resource-constrained schedules on a Gantt chart. The schedule used is the one produced in Example 4.19 using the CPM procedure. The output data set from PROC CPM is displayed in Output 8.13.2. Notice that the activities 'Drawings' and 'Mkt. Strat.' have been split to produce a shorter project duration than if they had not been split.

PROC GANTT is invoked with all default options and an ID statement. The early/late schedule is drawn on the first bar, and the resource-constrained schedule is drawn on the second bar of each activity. The observations corresponding to the split segments of each activity have been combined to produce the plot of the resource-constrained schedule for that activity. Thus, even though the Schedule data set input to PROC GANTT contains 18 observations, the Gantt chart shows each of the 14 activities only once.

```
title h=1.75 'Gantt Example 13';
title2 h=1.25 'Resource Constrained Schedule';

* set vpos to 50 and hpos to 100;
goptions vpos=50 hpos=100;
```

```
* plot the resource-constrained schedule using proc gantt;
proc gantt data=spltschd holidata=holdata;
   chart / holiday=(hol) dur=days
           height=1.6 pcompress;
   id task;
   run;
```

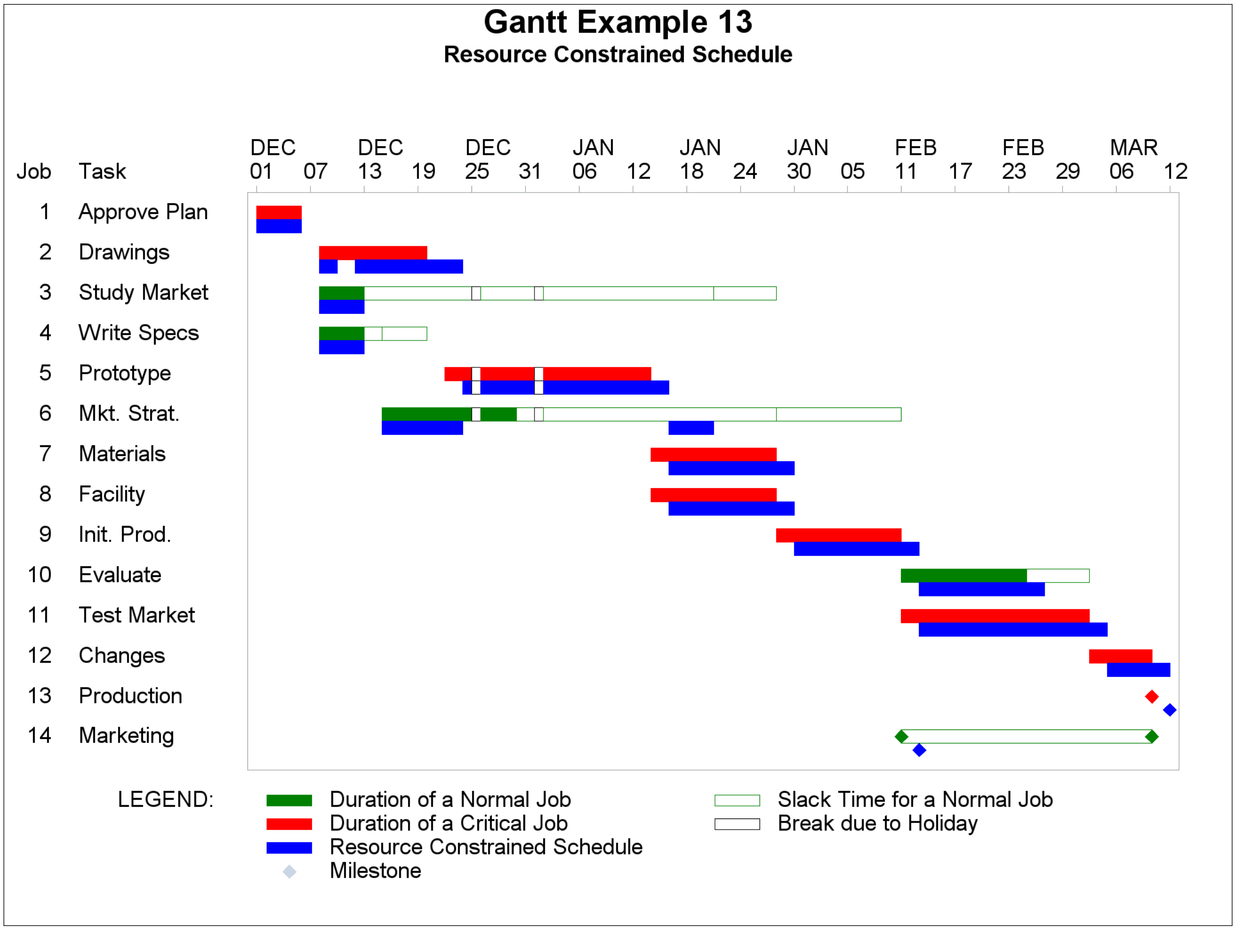
Output 8.13.1 Schedule Data Set SPLTSCHD

Gantt Example 13
Project Schedule: Splitting Allowed

Obs	Task	succ	SEGMT_NO	days	prodman	hrdware
1	Approve Plan	Drawings	.	5	1	.
2	Drawings	Prototype	.	10	.	1
3	Drawings	Prototype	1	2	.	1
4	Drawings	Prototype	2	8	.	1
5	Study Market	Mkt. Strat.	.	5	.	.
6	Write Specs	Prototype	.	5	.	.
7	Prototype	Materials	.	15	1	.
8	Mkt. Strat.	Test Market	.	10	1	.
9	Mkt. Strat.	Test Market	1	7	1	.
10	Mkt. Strat.	Test Market	2	3	1	.
11	Materials	Init. Prod.	.	10	.	.
12	Facility	Init. Prod.	.	10	.	.
13	Init. Prod.	Test Market	.	10	1	.
14	Evaluate	Changes	.	10	1	.
15	Test Market	Changes	.	15	.	.
16	Changes	Production	.	5	.	.
17	Production		.	0	1	.
18	Marketing		.	0	.	.

Obs	S_START	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
1	01DEC03	05DEC03	01DEC03	05DEC03	01DEC03	05DEC03
2	08DEC03	23DEC03	08DEC03	19DEC03	08DEC03	19DEC03
3	08DEC03	09DEC03	08DEC03	19DEC03	08DEC03	19DEC03
4	12DEC03	23DEC03	08DEC03	19DEC03	08DEC03	19DEC03
5	08DEC03	12DEC03	08DEC03	12DEC03	21JAN04	27JAN04
6	08DEC03	12DEC03	08DEC03	12DEC03	15DEC03	19DEC03
7	24DEC03	15JAN04	22DEC03	13JAN04	22DEC03	13JAN04
8	15DEC03	20JAN04	15DEC03	29DEC03	28JAN04	10FEB04
9	15DEC03	23DEC03	15DEC03	29DEC03	28JAN04	10FEB04
10	16JAN04	20JAN04	15DEC03	29DEC03	28JAN04	10FEB04
11	16JAN04	29JAN04	14JAN04	27JAN04	14JAN04	27JAN04
12	16JAN04	29JAN04	14JAN04	27JAN04	14JAN04	27JAN04
13	30JAN04	12FEB04	28JAN04	10FEB04	28JAN04	10FEB04
14	13FEB04	26FEB04	11FEB04	24FEB04	18FEB04	02MAR04
15	13FEB04	04MAR04	11FEB04	02MAR04	11FEB04	02MAR04
16	05MAR04	11MAR04	03MAR04	09MAR04	03MAR04	09MAR04
17	12MAR04	12MAR04	10MAR04	10MAR04	10MAR04	10MAR04
18	13FEB04	13FEB04	11FEB04	11FEB04	10MAR04	10MAR04

Output 8.13.2 Plotting the Resource-Constrained Schedule



Example 8.14: Specifying the Schedule Data Directly

Although each of the examples shown so far uses PROC CPM to produce the Schedule data set for PROC GANTT, this is by no means a requirement of the GANTT procedure. While the CPM procedure is a convenient means for producing different types of schedules, you can create your own schedule and draw a Gantt chart of the schedule without any intervention from PROC CPM. This is done by storing the schedule information in a SAS data set and specifying the data set name using the DATA= option in the PROC GANTT statement. It is also not necessary for the variables in the data set to have specific names, although giving the variables certain names can eliminate the need to explicitly identify them in the CHART statement.

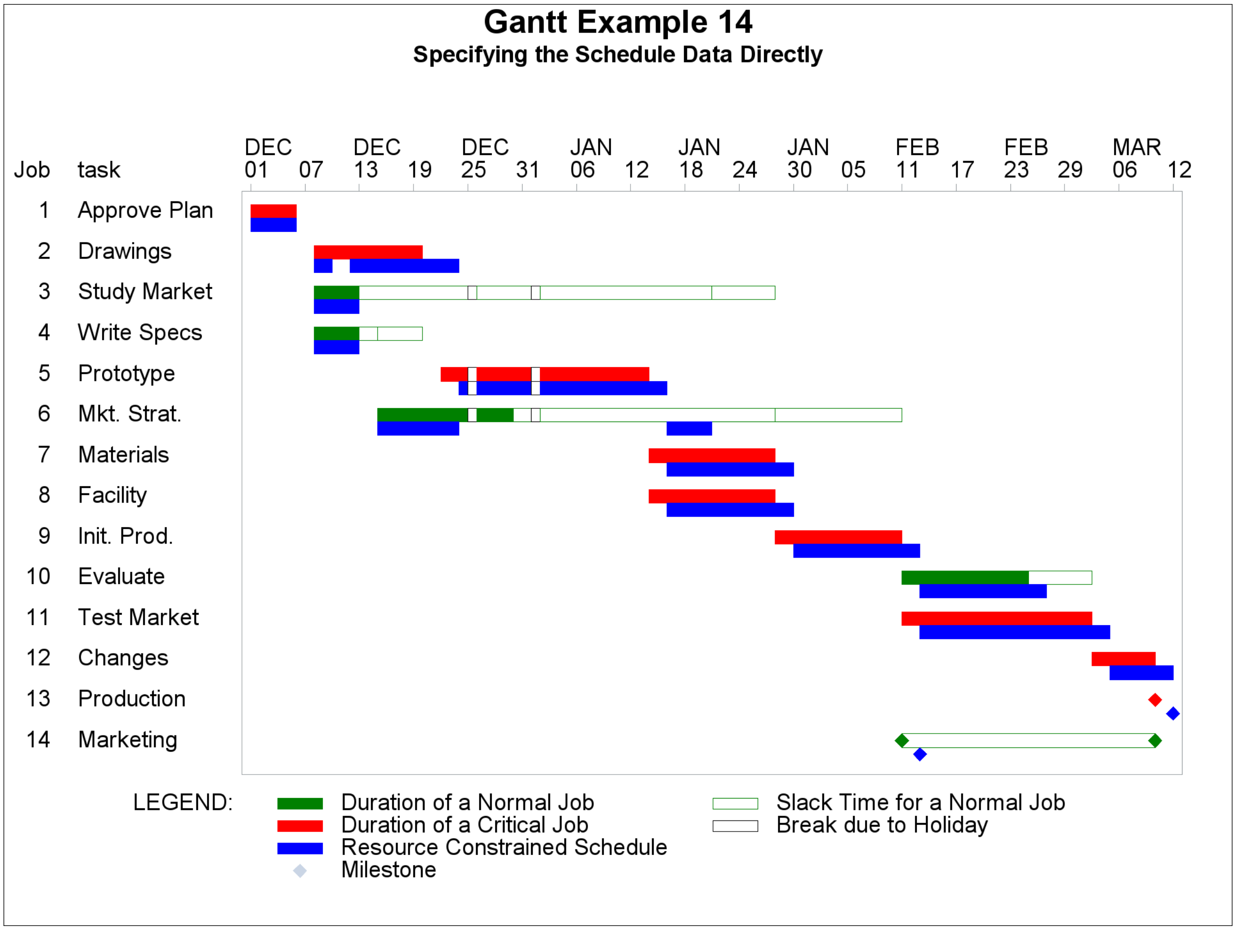
An example of the direct type of input can be seen in Example 8.10 which illustrates plotting of the actual schedule. In Example 8.10, PROC CPM was used to compute the predicted early/late schedule, which was then stored in the SAVEH data set. However, information about the actual schedule, which was provided in the COMPLETE data set, was not used by PROC CPM. Instead, this information was merged with the SAVEH data set to form WIDGELA, the Schedule data set for PROC GANTT. The variables representing the actual start and finish were identified to PROC GANTT using the A_START= and A_FINISH= options, respectively, in the CHART statement. The identification of the variables would not have been necessary if the start and finish variable names were A_START and A_FINISH, respectively.

The following example draws a Gantt chart of the early, late, and resource-constrained schedules for the widget manufacturing project. The schedule information is held in the WIDGDIR data set. The WIDGDIR data set contains the variables TASK, SEGMT_NO, DUR, RS, RF, E_START, E_FINISH, SDATE, and FDATE. The variable TASK identifies the activity. E_START and E_FINISH are recognized as the default names of the early start and early finish variables, respectively. The variables SDATE and FDATE define the late start and late finish times, respectively. Since these are not the default names for the late schedule variables, they need to be identified as such by specifying the LS= and LF= options (or the L_START= and L_FINISH= options) in the CHART statement. The variables RS and RF represent the resource-constrained start and finish times, respectively. As with the late schedule, these variables need to be identified to PROC GANTT by specifying the SS= and SF= options (or the S_START= and S_FINISH= options) in the CHART statement. Further, the SEGMT_NO variable identifies the segment number of the resource constrained schedule that an observation corresponds to since these are activities that start and stop multiple times before completion. The ZDUR variable is identified as a zero duration indicator by specifying the DUR= option in the CHART statement. Since ZDUR is zero for 'Production' and 'Marketing,' these activities are represented by milestones on the chart. Notice that although all the other activities have a value of '1' for the ZDUR variable, any nonzero value will produce the same result. This is due to the fact that PROC GANTT only uses this variable as an *indicator* of whether the activity has zero duration or not, in contrast to the interpretation of the DURATION variable in PROC CPM.

```
options ps=60 ls=100;

title h=1.75 'Gantt Example 14';

 /* Activity-on-Node representation of the project */
data widgdir;
   format task $12. rs rf e_start e_finish sdate fdate date7.;
   input task & segmt_no zdur rs & date7. rf & date7.
         e_start & date7. e_finish & date7.
         sdate & date7.  fdate & date7.;
   datalines;
Approve Plan  . 1 01DEC03  05DEC03  01DEC03  05DEC03  01DEC03  05DEC03
Drawings      . 1 08DEC03  23DEC03  08DEC03  19DEC03  08DEC03  19DEC03
Drawings     1 1 08DEC03  09DEC03  08DEC03  19DEC03  08DEC03  19DEC03
Drawings     2 1 12DEC03  23DEC03  08DEC03  19DEC03  08DEC03  19DEC03
Study Market  . 1 08DEC03  12DEC03  08DEC03  12DEC03  21JAN04  27JAN04
Write Specs   . 1 08DEC03  12DEC03  08DEC03  12DEC03  15DEC03  19DEC03
Prototype     . 1 24DEC03  15JAN04  22DEC03  13JAN04  22DEC03  13JAN04
Mkt. Strat.   . 1 15DEC03  20JAN04  15DEC03  29DEC03  28JAN04  10FEB04
Mkt. Strat.  1 1 15DEC03  23DEC03  15DEC03  29DEC03  28JAN04  10FEB04
Mkt. Strat.  2 1 16JAN04  20JAN04  15DEC03  29DEC03  28JAN04  10FEB04
Materials     . 1 16JAN04  29JAN04  14JAN04  27JAN04  14JAN04  27JAN04
Facility      . 1 16JAN04  29JAN04  14JAN04  27JAN04  14JAN04  27JAN04
Init. Prod.   . 1 30JAN04  12FEB04  28JAN04  10FEB04  28JAN04  10FEB04
Evaluate      . 1 13FEB04  26FEB04  11FEB04  24FEB04  18FEB04  02MAR04
Test Market   . 1 13FEB04  04MAR04  11FEB04  02MAR04  11FEB04  02MAR04
Changes       . 1 05MAR04  11MAR04  03MAR04  09MAR04  03MAR04  09MAR04
Production    . 0 12MAR04  12MAR04  10MAR04  10MAR04  10MAR04  10MAR04
Marketing     . 0 13FEB04  13FEB04  11FEB04  11FEB04  10MAR04  10MAR04
;
```

```
data holdata;
   format hol date7.;
   input hol & date7.;
   datalines;
25dec03
01jan04
;

/* set up required pattern statements */
pattern1 c=green v=s;    /* duration of a non-critical activity  */
pattern2 c=green v=e;    /* slack time for a noncrit. activity   */
pattern3 c=red v=s;      /* duration of a critical activity      */
pattern4 c=magenta v=e;  /* slack time for a supercrit. activity */
pattern5 c=magenta v=s;  /* duration of a supercrit. activity    */
pattern6 c=cyan v=s;     /* actual duration of an activity       */
pattern7 c=black v=e;    /* break due to a holiday               */
pattern8 c=blue v=s;     /* resource schedule of activity        */
pattern9 c=brown v=s;    /* baseline schedule of activity        */

title2 h=1.25 'Specifying the Schedule Data Directly';

proc gantt data=widgdir holidata=holdata;
   chart / holiday=(hol) dur=zdur
           ss=rs sf=rf ls=sdate lf=fdate
           height=1.5 pcompress;
   id task;
   run;
```

Output 8.14.1 Specifying the Schedule Data Directly

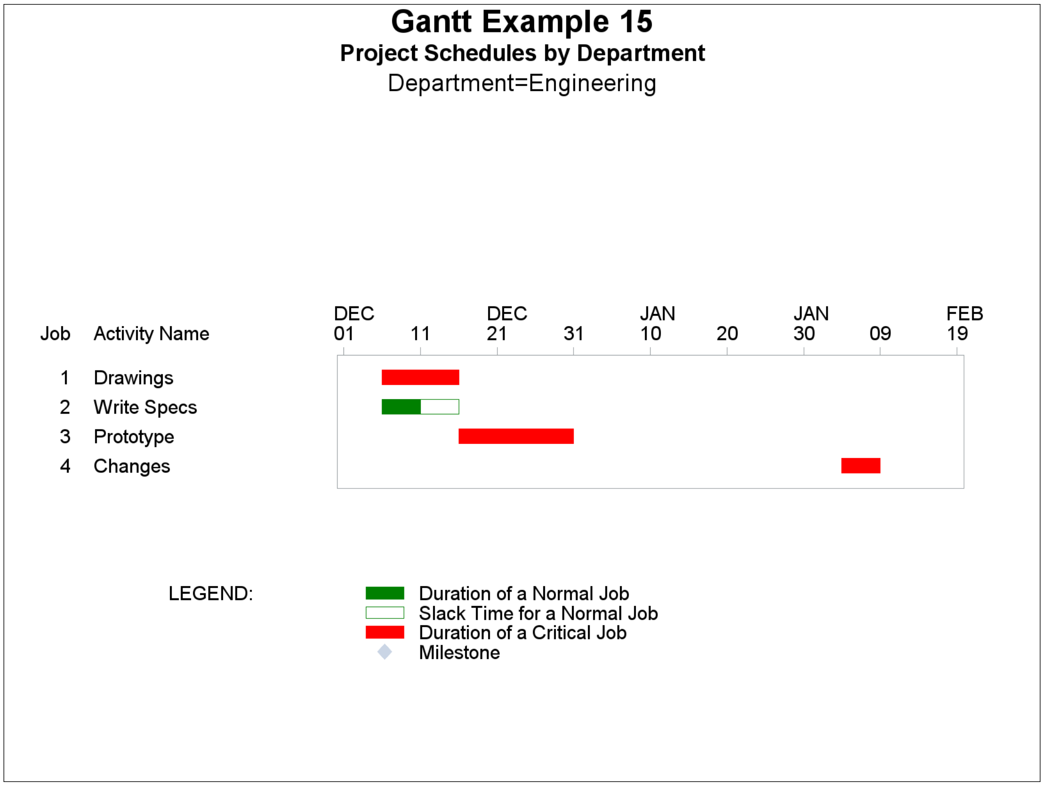


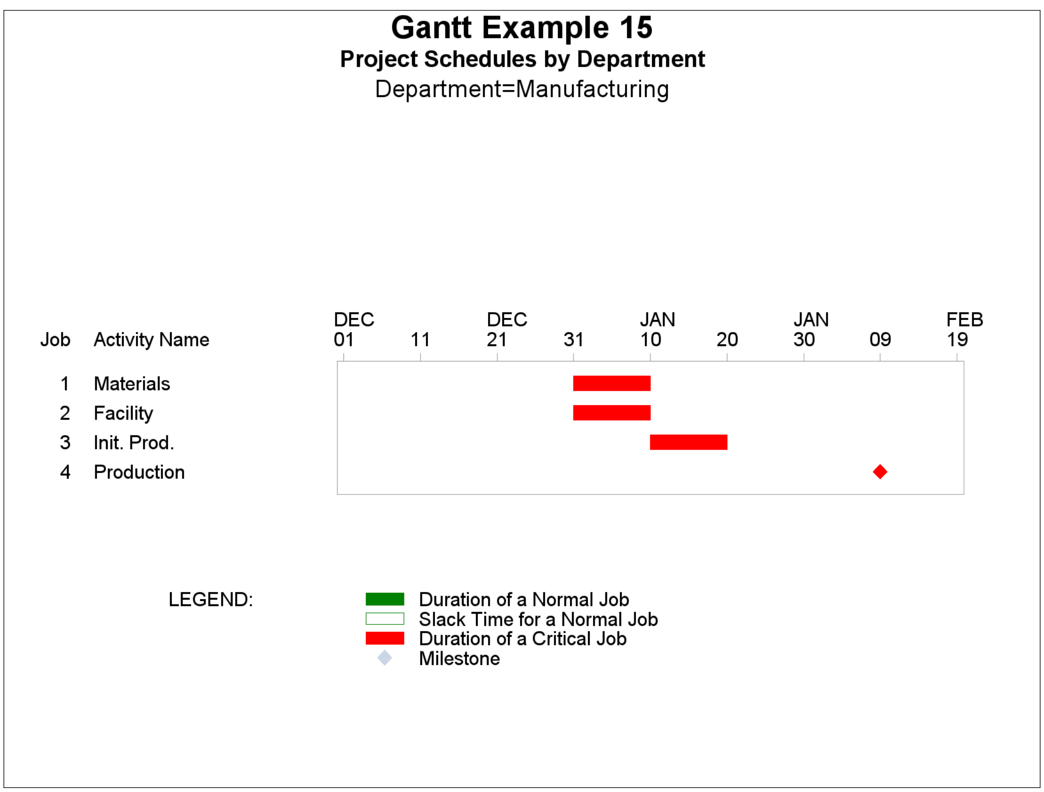
Example 8.15: BY Processing

Every activity in the widget manufacturing project is carried out by one of five departments: Planning, Engineering, Marketing, Manufacturing, and Testing. The DETAILS data set in Example 8.6 identifies the department responsible for each activity. Thus, the project can be thought of as made up of five smaller subprojects, a subproject being the work carried out by a department. A foreseeable need of the project manager and every department is a separate Gantt chart for each subproject. This example uses the WIDGETN data set from Example 4.1, which is formed by merging the WIDGET data set with the DETAILS data set. After scheduling the master project using PROC CPM with DEPT as an ID variable, the Schedule data set is sorted by department name and early start time. The GANTT procedure is then invoked with the variable DEPT specified in the BY statement to obtain individual Gantt charts for each subproject. The Gantt charts for the five different subprojects are shown in Output 8.15.1. The MINDATE= and MAXDATE= options have been specified to ensure a consistent date range across projects. Notice that the TITLE2 statement uses the text substitution option #BYVAR*n*, which substitutes the name of the *n*th BY variable. The BY-LINE that appears below the titles identifies the current values of the BY variables. You can suppress this using the NOBYLINE option in an OPTION statement or the HBY option in a GOPTIONS statement. The SPLIT= option is specified to prevent the TASK variable label from being split on the embedded blank.

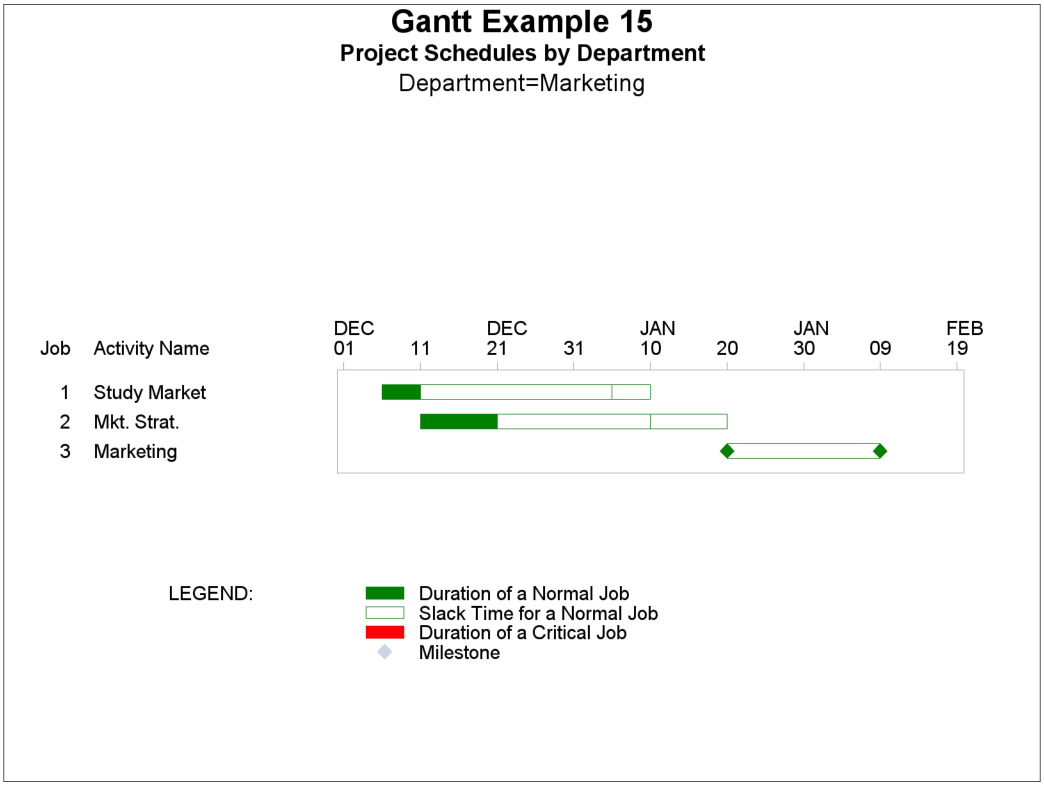
```
data widgetn;
   label task = "Activity Name";
   merge widget details;
   run;

proc cpm date='01dec03'd data=widgetn;
   activity task;
   duration days;
   successor succ1 succ2 succ3;
   id dept;
   run;

proc sort;
   by dept e_start;
   run;


proc gantt split='/';
   chart / pcompress scale=1 dur=days height=1.2
           mindate='01dec03'd maxdate='11feb04'd;
   by dept;
   id task;
   run;
```

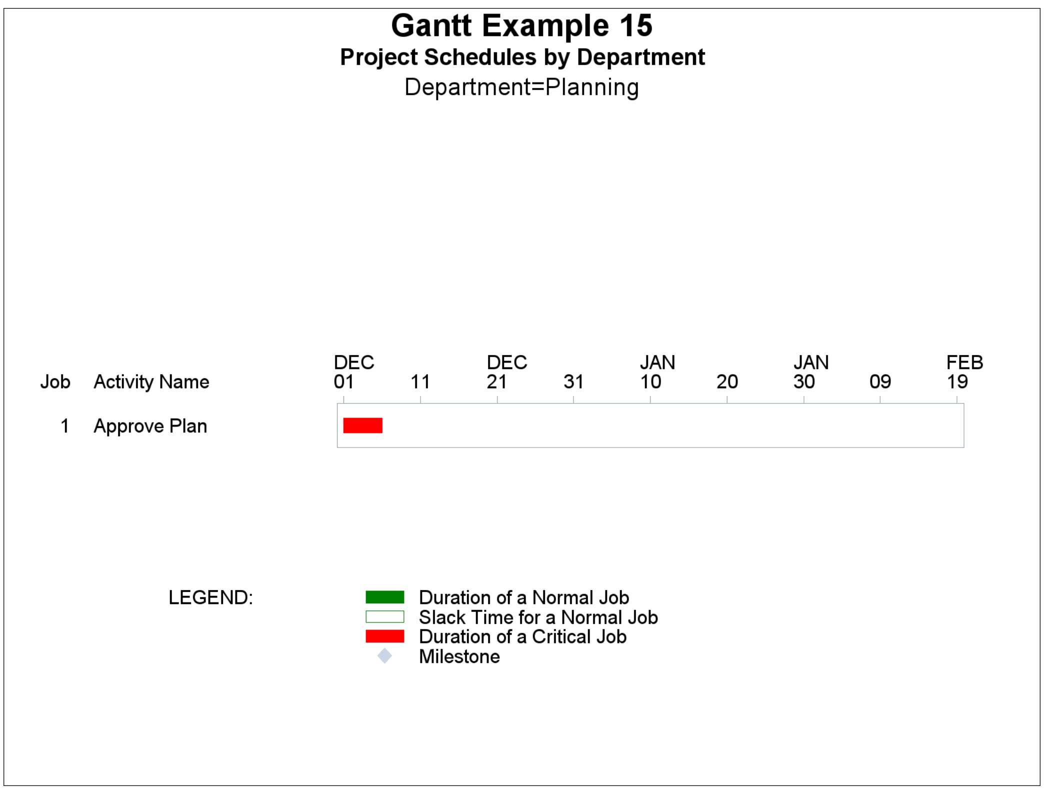
Output 8.15.1 Using BY Processing for Separate Gantt Charts



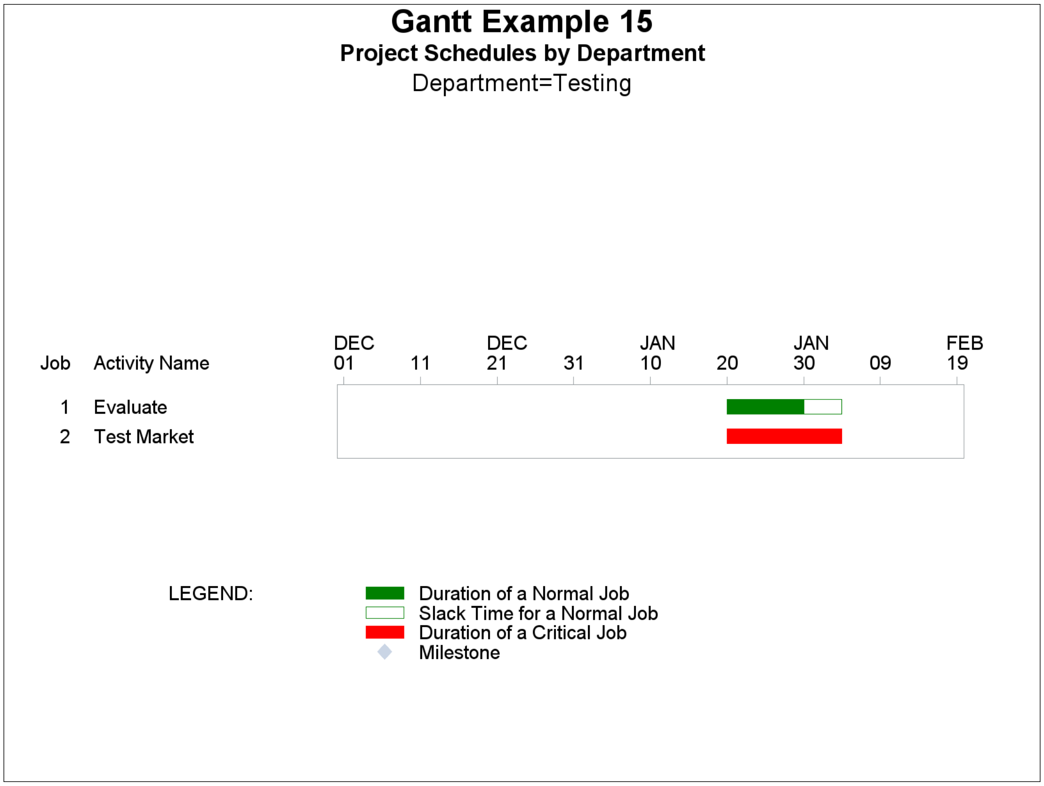


Output 8.15.1 *continued*





Output 8.15.1 *continued*



Example 8.16: Gantt Charts by Persons

Now suppose that you want to obtain individual Gantt charts for two people (Thomas and William) working on the widget manufacturing project. The data set WIDGBYGP, displayed in Output 8.16.1, contains two new variables, THOMAS and WILLIAM. Each variable has a value '1' for activities in which the person is involved and a missing value otherwise. Thus, a value of '1' for the variable THOMAS in observation number 2 indicates that Thomas is working on the activity 'Drawings.'

PROC CPM is used to schedule the project to start on December 1, 2003. A data set named PERSONS is created containing one observation per activity per person working on that activity and a new variable named PERSON containing the name of the person to which the observation pertains. For example, this new data set contains two observations for the activity 'Write Specs,' one with PERSON='Thomas' and the other with PERSON='William,' and no observation for the activity 'Approve Plan.' This data set is sorted by PERSON and E_START, and displayed in Output 8.16.2. PROC GANTT is next invoked with a BY statement to obtain individual charts for each person. The resulting Gantt charts are shown in Output 8.16.3. The BY-LINE is suppressed by specifying the NOBYLINE option in an OPTIONS statement and the name of the person corresponding to the chart is displayed in the subtitle by using the #BYVAL substitution in the TITLE2 statement.

Output 8.16.1 Data Set WIDGBYGP

Data widgbyp

Obs	task	days	tail	head	thomas	william
1	Approve Plan	5	1	2	.	.
2	Drawings	10	2	3	1	.
3	Study Market	5	2	4	.	.
4	Write Specs	5	2	3	1	1
5	Prototype	15	3	5	1	1
6	Mkt. Strat.	10	4	6	.	.
7	Materials	10	5	7	.	1
8	Facility	10	5	7	.	1
9	Init. Prod.	10	7	8	1	.
10	Evaluate	10	8	9	1	1
11	Test Market	15	6	9	.	.
12	Changes	5	9	10	1	.
13	Production	0	10	11	.	1
14	Marketing	0	6	12	.	.
15	Dummy	0	8	6	.	.

```
title h=1.75 'Gantt Example 16';

proc cpm data=widgbyp date='1dec03'd;
   tailnode tail;
   duration days;
   headnode head;
   id task thomas william;
run;

data persons;
   set _last_;
   if william^=. then do;
      person='William';
      output;
      end;
   if thomas^=. then do;
      person='Thomas';
      output;
      end;
   drop thomas william;
run;

proc sort data=persons;
   by person e_start;
run;

title2 'Data PERSONS';
proc print data=persons;
   run;
```

```
/* suppress byline */
options nobyline;

goptions hpos=120 vpos=40 htext=1.1;

title2 h=1.25 'Personalized Gantt Chart for #BYVAL(person)';
proc gantt data=persons;
   chart / pcompress;
   by person;
   id task;
   run;
```

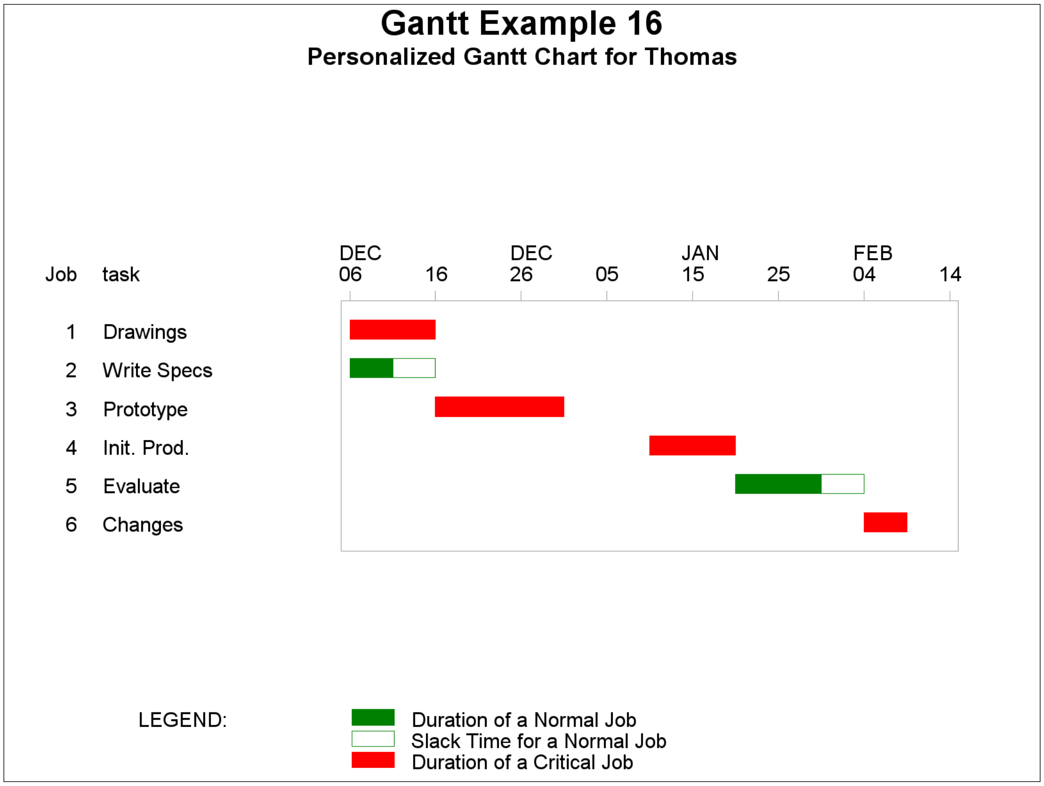
Output 8.16.2 Data Set PERSONS

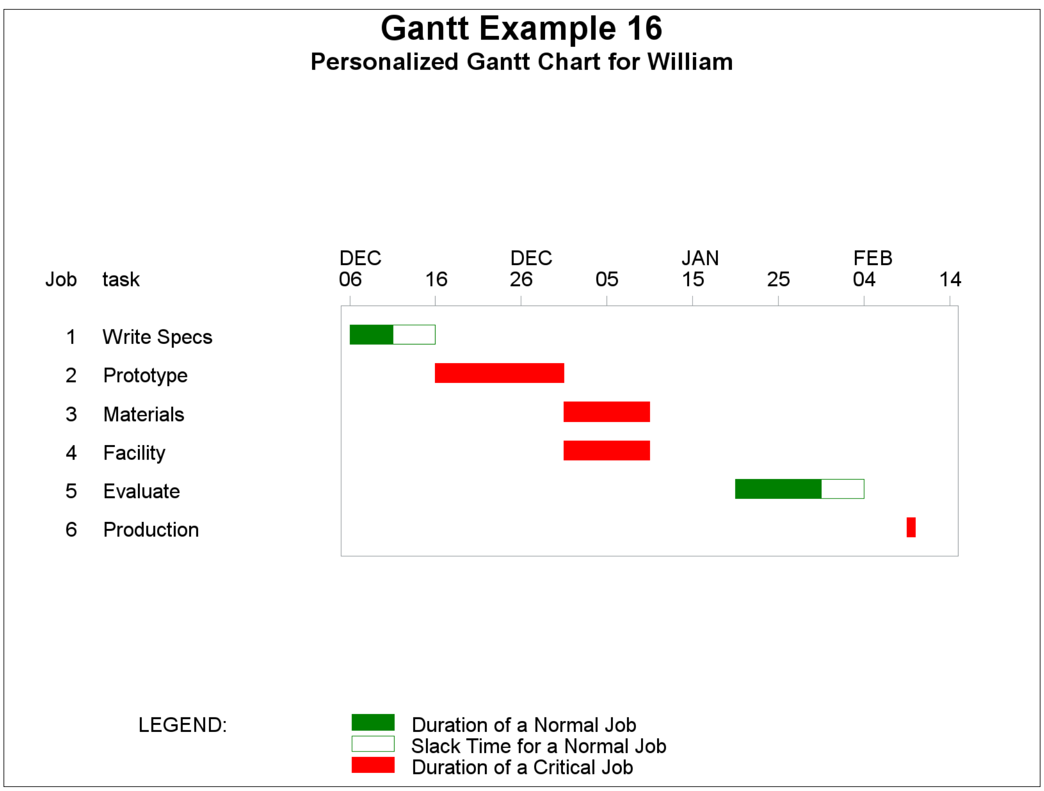
Gantt Example 16

Data PERSONS

Obs	tail	head	days	task	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT	person
1	2	3	10	Drawings	06DEC03	15DEC03	06DEC03	15DEC03	0	0	Thomas
2	2	3	5	Write Specs	06DEC03	10DEC03	11DEC03	15DEC03	5	5	Thomas
3	3	5	15	Prototype	16DEC03	30DEC03	16DEC03	30DEC03	0	0	Thomas
4	7	8	10	Init. Prod.	10JAN04	19JAN04	10JAN04	19JAN04	0	0	Thomas
5	8	9	10	Evaluate	20JAN04	29JAN04	25JAN04	03FEB04	5	5	Thomas
6	9	10	5	Changes	04FEB04	08FEB04	04FEB04	08FEB04	0	0	Thomas
7	2	3	5	Write Specs	06DEC03	10DEC03	11DEC03	15DEC03	5	5	William
8	3	5	15	Prototype	16DEC03	30DEC03	16DEC03	30DEC03	0	0	William
9	5	7	10	Materials	31DEC03	09JAN04	31DEC03	09JAN04	0	0	William
10	5	7	10	Facility	31DEC03	09JAN04	31DEC03	09JAN04	0	0	William
11	8	9	10	Evaluate	20JAN04	29JAN04	25JAN04	03FEB04	5	5	William
12	10	11	0	Production	09FEB04	09FEB04	09FEB04	09FEB04	0	0	William

Output 8.16.3 Gantt Charts by Person



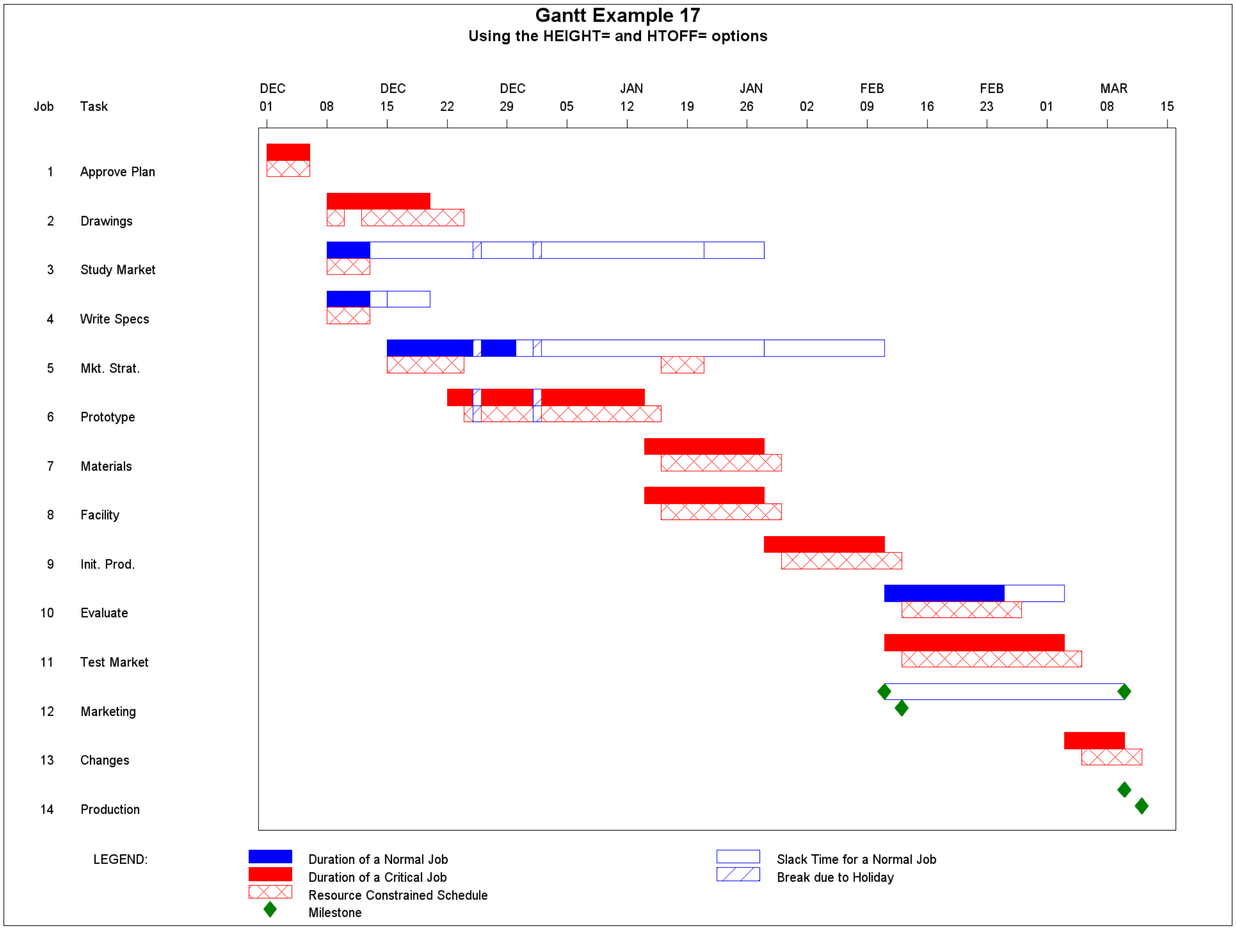


Example 8.17: Using the HEIGHT= and HTOFF= Options

The following example illustrates two options that control the height and positioning of all text produced by PROC GANTT. The data used for this example come from Example 8.13, which illustrates plotting of the resource-constrained schedule. PATTERN statements are specified in order to identify the fill patterns for the different schedule types and holidays. The resource-constrained schedule is drawn using the fill pattern from the eighth PATTERN statement. The HEIGHT= option is set to 2, indicating that the height of all text produced by PROC GANTT be equal to the height of two activity bars. This text includes activity text, legend text, and axis labeling text. The HTOFF= option is also set to 2, which drops the font baseline of the activity text by the height of one schedule bar causing the font baseline to be positioned at the bottom of the resource-constrained schedule bar. The resulting Gantt chart is displayed in Output 8.17.1.

```
title 'Gantt Example 17';

* set up required pattern statements;

pattern1 c=blue v=s;   /* duration of a non-critical activity   */
pattern2 c=blue v=e;   /* slack time for a noncrit. activity    */
pattern3 c=red v=s;    /* duration of a critical activity       */
pattern4 c=red v=e;    /* slack time for a supercrit. activity  */
pattern5 c=red v=r2;   /* duration of a supercrit. activity     */
pattern6 c=cyan v=s;   /* actual duration of an activity        */
pattern7 c=blue v=r1;  /* break due to a holiday                */
pattern8 c=red v=x1;   /* resource schedule of activity         */
pattern9 c=blue v=s;   /* baseline schedule of activity         */

* set vpos to 50 and hpos to 100;
goptions vpos=50 hpos=100;

title2 'Using the HEIGHT= and HTOFF= options';

* draw Gantt chart using height and htoff equal to 2;
proc gantt graphics data=spltschd holidata=holdata;
   chart / holiday=(hol) dur=days compress cmile=green caxis=black
           height=2 htoff=2;
   id task;
   run;
```

Output 8.17.1 Using the HEIGHT= and HTOFF= options

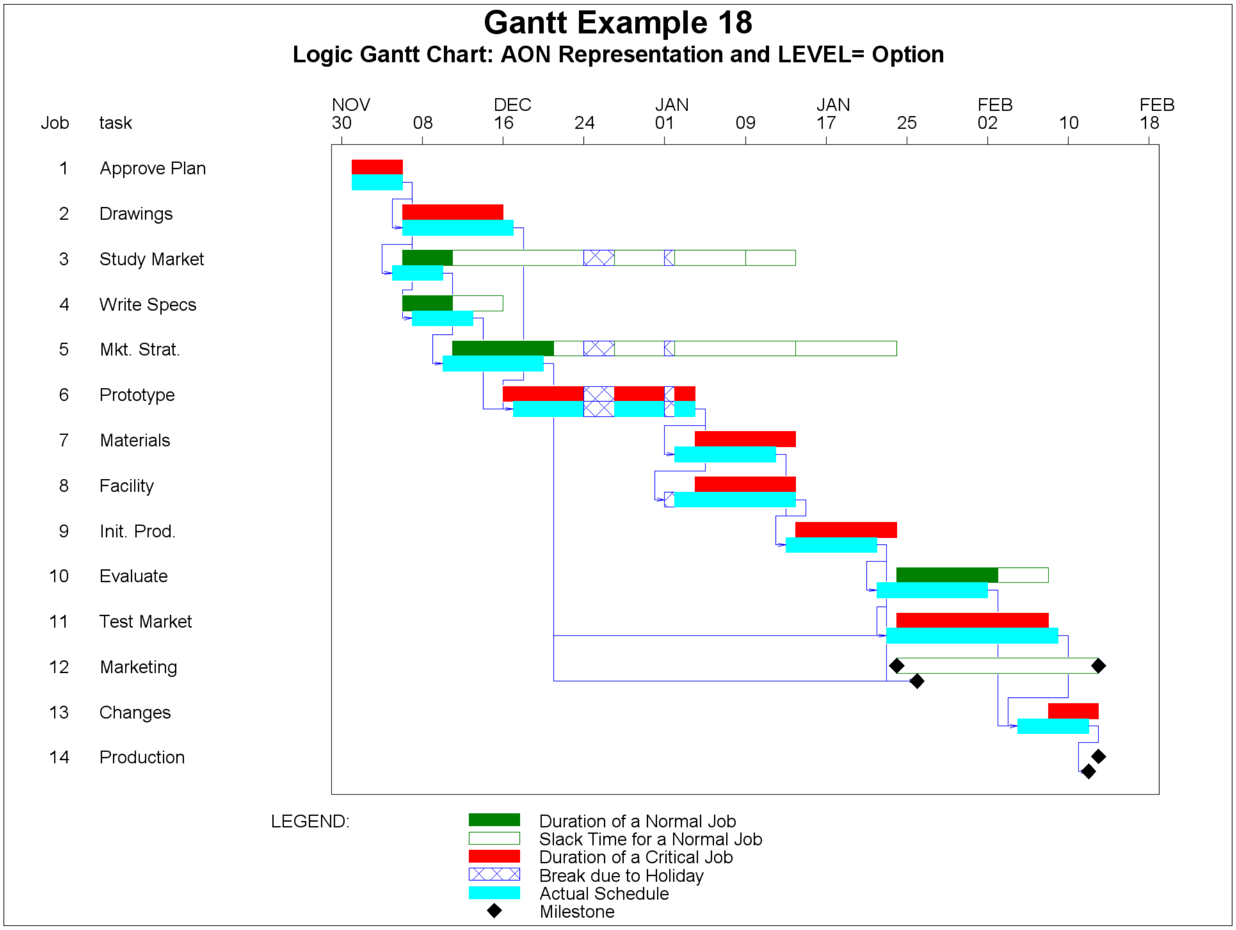


Example 8.18: Drawing a Logic Gantt Chart Using AON Representation

This example uses the data of Example 8.10, which illustrates the drawing of the actual schedule. The ACTIVITY= and SUCCESSOR= options are specified in the CHART statement to define the precedence relationships using the AON format to PROC GANTT. Since no LAG= option is specified, the lag type of each connection is assumed to be Finish-to-Start (FS). In this case, the precedence defining variables exist in the WIDGELA data set; however, this is not a requirement. The precedence defining variables can belong to a different data set as long as the ACTIVITY variable is common to both data sets and the PRECDATA= option, identifying the Precedence data set, is specified in the PROC GANTT statement. Setting the LEVEL= option to 2 causes the actual schedule bar to be used as the logic bar; that is, PROC GANTT draws the precedence connections with respect to the actual schedule. By default, the precedence connections are drawn with respect to the first bar. The color of the precedence connections is specified with the CPREC= option in the CHART statement. You can change the line style and line width of the precedence connections by specifying the LPREC= and WPREC= options in the CHART statement. The resulting Gantt chart is shown in Output 8.18.1.

```
title h=1.75 'Gantt Example 18';

title2 h=1.25 'Logic Gantt Chart: AON Representation and LEVEL= Option';

* sort the data;
proc sort;
   by e_start;
   run;

* set graphics options;
goptions vpos=50 hpos=100 htext=1.2;

* draw the logic Gantt chart;
proc gantt graphics data=widgela holidata=holidays;
   chart / holiday=(holiday) holifin=(holifin)
           a_start=sdate a_finish=fdate dur=days
           compress
           cmile=black
           activity=task successor=(succ1-succ3)
           caxis=black
           level=2
           cprec=blue;
   id task;
   run;
```

Output 8.18.1 Drawing a Logic Gantt Chart Using AON Representation



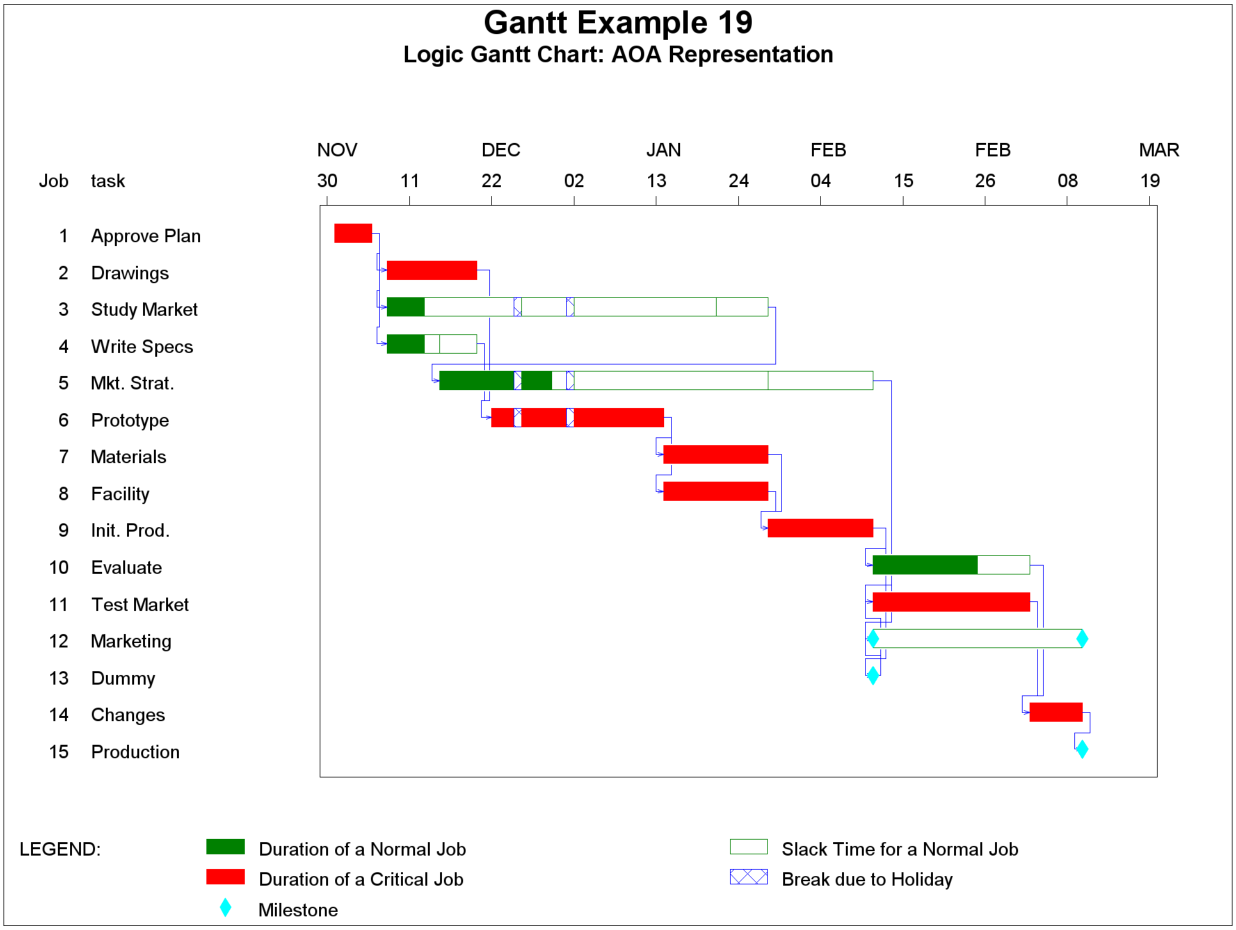
Example 8.19: Specifying the Logic Control Options

This example illustrates four options that control the routing of a precedence connection from an activity to its successor on the logic Gantt chart. The example also illustrates the drawing of a Logic Gantt chart using the Activity-on-Arc format.

The Activity data set for PROC CPM is the WIDGETA data set from Example 4.2, which defines the widget manufacturing project in AOA format. The project is scheduled subject to weekends, and the holidays are defined in the HOLDATA data set. The resulting schedule is stored in the output data set SAVEHP. The GANTT procedure is next invoked to produce a Logic Gantt chart by specifying the HEAD= and TAIL= options in the CHART statement. The resulting Logic Gantt chart is shown in Output 8.19.1.

```
title h=1.75 'Gantt Example 19';

data holdata;
   format hol date7.;
   input hol & date7.;
   datalines;
25dec03
01jan04
;

* schedule the project subject to holidays and weekends;
proc cpm data=widgeta holidata=holdata out=savehp
         date='1dec03'd interval=weekday;
   tailnode tail;
   headnode head;
   duration days;
   holiday  hol;
   id task dept descrpt;
   run;

* sort the schedule by the early start date;
proc sort;
   by e_start;
   run;

* set background to white and text to black;
goptions cback=white ctext=black;

* set additional graphics options;
goptions vpos=50 hpos=100 htext=1.1;

 * plot the logic Gantt chart using AOA representation;
title2 h=1.25 'Logic Gantt Chart: AOA Representation';

proc gantt graphics data=savehp holidata=holdata;
   chart / compress cprec=blue caxis=black cmile=cyan
           increment=7 height=1.5
           dur=days holiday=(hol)
           head=head tail=tail;
   id task;
   run;
```

Output 8.19.1 Logic Gantt Chart: AOA Representation

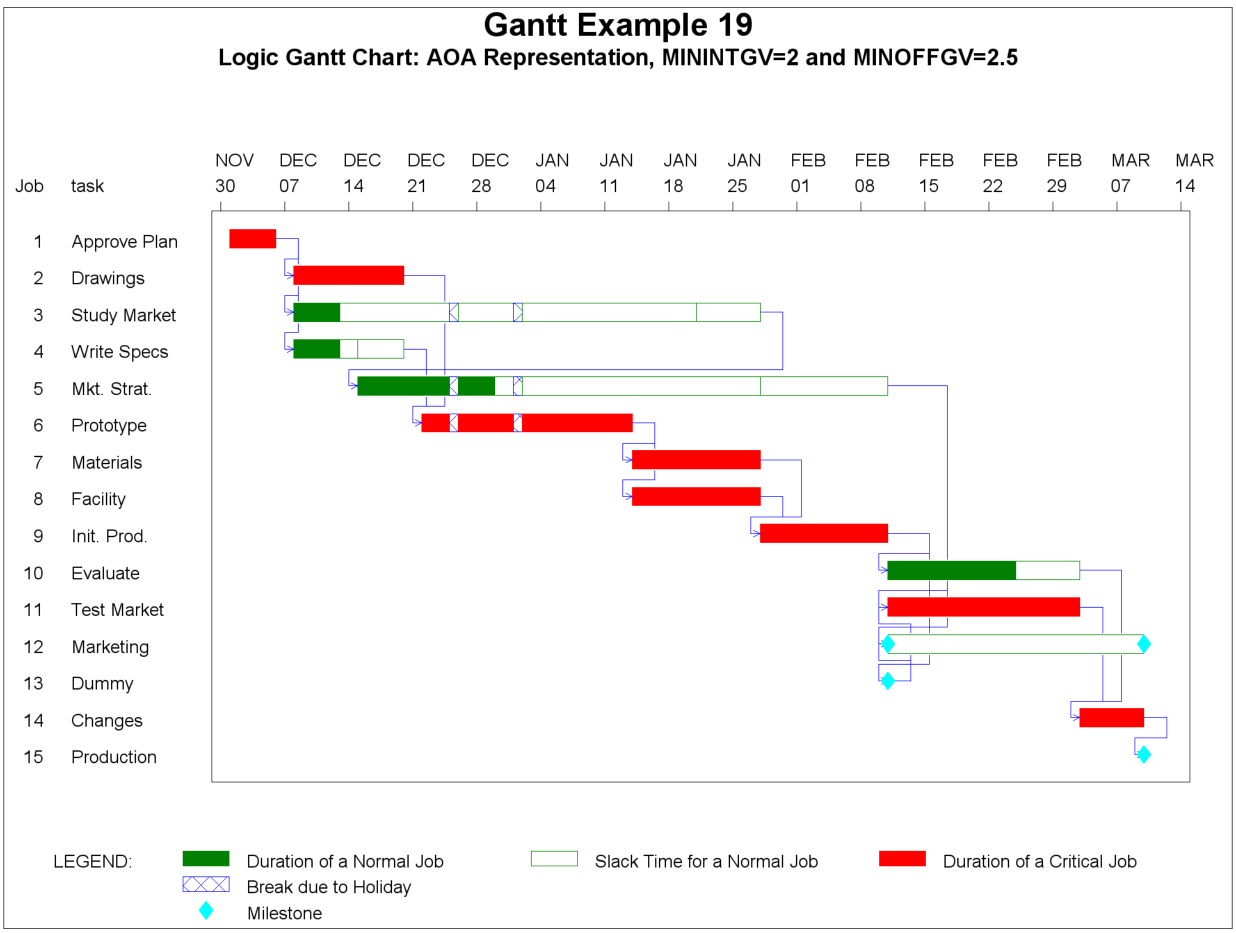


The next invocation of PROC GANTT illustrates the effect of the MININTGV= and MINOFFGV= options, which control placement of the global verticals. The concept of global verticals is explained in the section "Specifying the Logic Options" on page 546. The data sets from the previous invocation of the GANTT procedure remain unchanged. The minimum distance of a global vertical from the end of the bar it is associated with is increased from its default of 1 cell to 2.5 cells by specifying MINOFFGV=2.5. Likewise, the minimum distance between any two global verticals is increased from its default of .75 cells to 2 cells by specifying MININTGV=2.0. The effects of these changes are visible in the resulting Logic Gantt chart shown in Output 8.19.2.

```
goptions htext=1.4;

* illustrate the minintgv and minoffgv options;
title2 h=1.25
   'Logic Gantt Chart: AOA Representation, MININTGV=2 and MINOFFGV=2.5';

proc gantt graphics data=savehp holidata=holdata;
   chart / dur=days holiday=(hol) compress increment=7
           cprec=blue caxis=black cmile=cyan
           head=head tail=tail
           minintgv=2.0 minoffgv=2.5;
   id task;
   run;
```

Notice that now there is greater distance between vertical segments (corresponding to global verticals), and the horizontal segments leaving bars are longer.

Output 8.19.2 Specifying the MININTGV= and MINOFFGV= Options

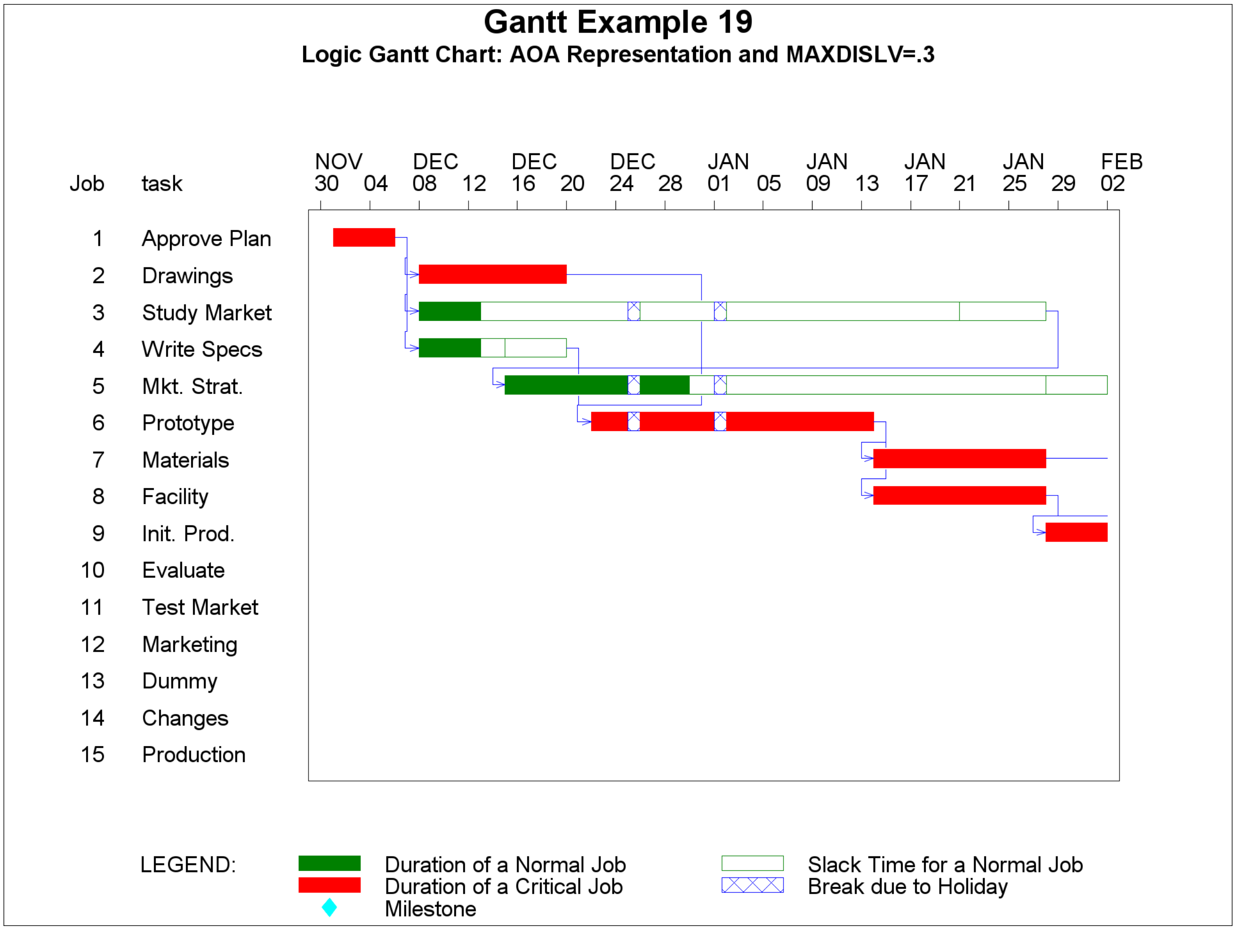


The MAXDATE= option is specified in the remaining Gantt calls in this example in order to focus on the schedule bars of the first few activities in the chart. The next two outputs illustrate the use of the MAXDISLV= option in the CHART statement. The MAXDISLV= option is used as a safeguard to limit the feasible region made available to PROC GANTT for placement of local verticals. The value specified dictates the maximum allowable displacement of the local vertical from its ideal position, that is, at a distance of MINOFFLV= from the end of the bar with which it is associated. However, this ideal position may tend to be positioned too close to a global vertical or even coincide with one. Depending on the cell width, this can result in visual misinterpretation of the Logic Gantt chart. In order to avoid this scenario, you should specify a reasonable value for the MAXDISLV= option to permit a certain amount of freedom for local vertical placement so as to distinguish between local and global verticals. Typically, use of this option is desirable when the value of the MININTGV= option, the minimum distance between global verticals, is relatively much greater than the value of the MAXDISLV= option.

To illustrate, consider the following Gantt call with a large MININTGV= value (10) and a relatively smaller MAXDISLV= value (0.3). Thus, for every local vertical, PROC GANTT has a very small interval that is less than a third of a cell wide in which to place that local vertical regardless of whether a global vertical runs through that interval or not. The result of this constraint is illustrated in the chart shown in Output 8.19.3. The local vertical for ‘Drawings’ is positioned as far as possible from the global vertical of ‘Approve Plan,’ but the value of the MAXDISLV= option restricts it from being positioned any further. Visually it is not pleasing, and it is difficult to distinguish the local and global verticals. A similar situation is evident with the local vertical of ‘Prototype’ and the global vertical of ‘Write Specs.’

```
goptions htext=1.2;

* illustrate the maxdislv option;
title2 h=1.25 'Logic Gantt Chart: AOA Representation and MAXDISLV=.3';

proc gantt graphics data=savehp holidata=holdata;
   chart / compress cprec=blue caxis=black cmile=cyan
           dur=days holiday=(hol)
           head=head tail=tail
           maxdislv=.3 minintgv=10
           maxdate='01feb04'd;
   id task;
   run;
```

Output 8.19.3 Specifying the MAXDISLV= Option (I)

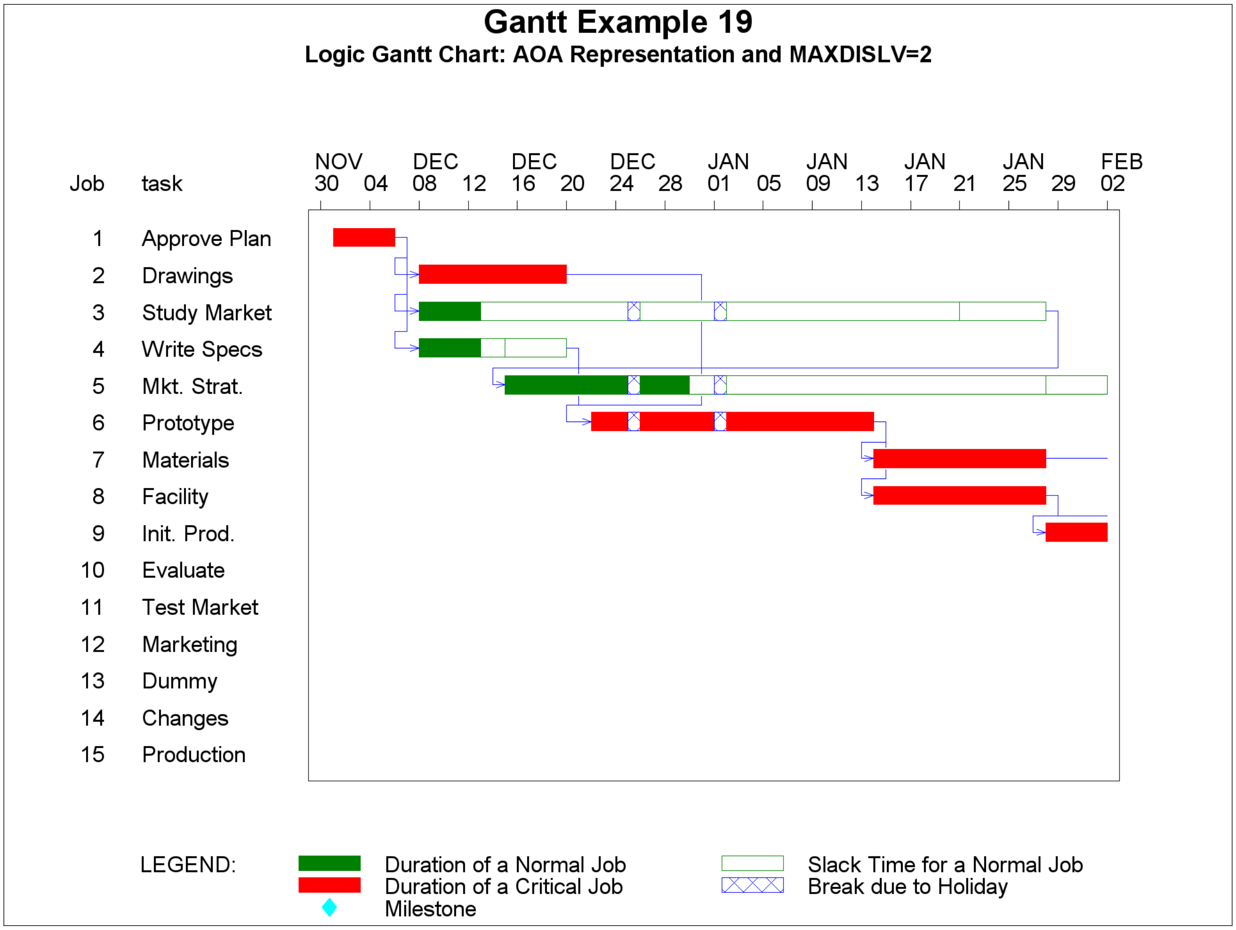


By reducing the value of MAXDISLV= even further, you can produce a chart that gives the appearance of a local vertical overlapping with a global vertical owing to resolution limitations of the display device. Theoretically, by design, this will never be the case. Recall that the value of the MAXDISLV= option is strictly positive and is at least one-tenth of a cell width.

The solution to this problem is to increase the value of the MAXDISLV= option so that the local vertical can be displaced further away from any adjacent global verticals. In the next invocation of PROC GANTT, the value of the MAXDISLV= option is increased to 2, resulting in a Logic Gantt chart in which the local verticals are staggered further away from nearby global verticals. This Gantt chart is displayed in Output 8.19.4.

```
title2 h=1.25 'Logic Gantt Chart: AOA Representation and MAXDISLV=2';

proc gantt graphics data=savehp holidata=holdata;
   chart / compress cprec=blue caxis=black cmile=cyan
           dur=days holiday=(hol)
           head=head tail=tail
           maxdislv=2 minintgv=10
           maxdate='01feb04'd;
   id task;
   run;
```

Output 8.19.4 Specifying the MAXDISLV= Option (II)

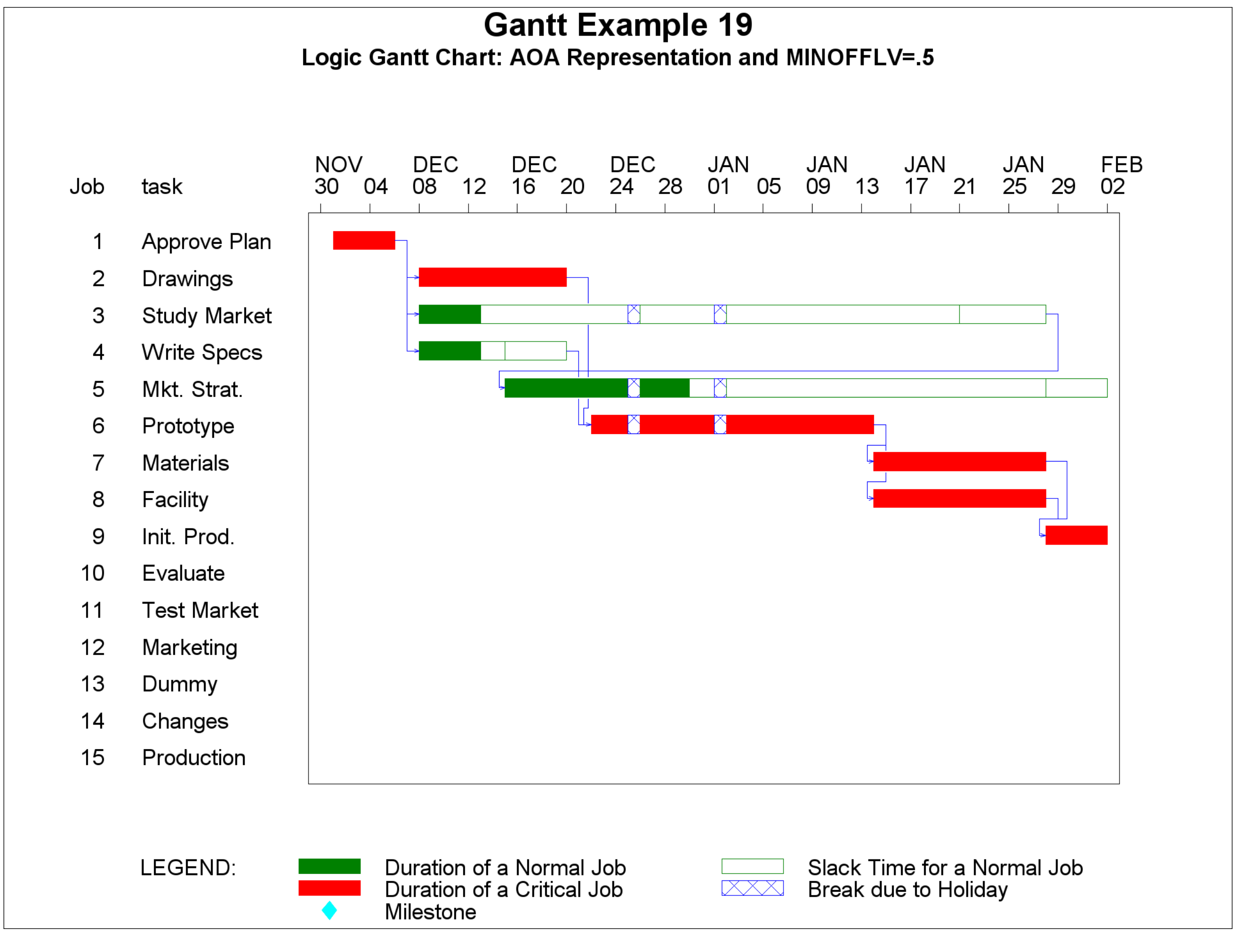


The final Gantt chart in this example illustrates the use of the MINOFFLV= option in the CHART statement. This option specifies the minimum distance of a local vertical from the end of the bar with which it is associated. Although the position corresponding to the MINOFFLV= option is the position of choice for placement of the local vertical, the actual placement can differ from this position owing to the presence of nearby global verticals, as illustrated by Output 8.19.3 and Output 8.19.4. The maximum amount of displacement is determined by the value of the MAXDISLV= option.

In all of the preceding charts in this example, the connection from the activity, 'Approve Plan,' to each of its three successors, 'Drawings', 'Study Market', and 'Write Specs', is a 5-segment connection similar to the type illustrated in Figure 8.14. This is caused by backtracking of the activity's global vertical to the successor's local vertical as described in the section "Controlling the Layout" on page 549. To transform this connection into a 3-segment connection as shown in Figure 8.13, you need to position the local vertical to the right of the global vertical. The following invocation of PROC GANTT achieves this by specifying MINOFFLV=0.5, and the resulting Gantt chart is shown in Output 8.19.5. Notice that this option affects the positioning of *all* local verticals on the chart in contrast to the MAXDISLV= option, which affects only those local verticals that are close to global verticals.

```
* illustrate the minofflv option;
title2 h=1.25
   'Logic Gantt Chart: AOA Representation and MINOFFLV=.5';

proc gantt graphics data=savehp holidata=holdata;
   chart / compress cprec=blue caxis=black cmile=cyan
           dur=days holiday=(hol)
           head=head tail=tail
           minofflv=.5
           maxdate='01feb04'd;
   id task;
   run;
```

Output 8.19.5 Specifying the MINOFFLV= Option



Example 8.20: Nonstandard Precedence Relationships

This example demonstrates the use of nonstandard precedence relationships and specification of the PRECDATA= option in the PROC GANTT statement.

The project and nonstandard precedence relationships are defined by the WIDGLAG2 data set, which is a modification of the WIDGLAG data set that was used in Example 4.11 to illustrate the CPM procedure. The activity and successor variables are represented by the TASK and SUCC variables, respectively, and the lag type of the relationship is defined by the LAGDUR variable. The LAGDUR variable defines the lag type in *keyword_duration_calendar* format for the purpose of passing the information to PROC CPM. Although PROC GANTT accepts this format for a lag variable, it does not use the *duration* and *calendar* values when drawing the connection since the schedule is already computed at this time (presumably by PROC CPM).

As in the WIDGLAG data set, the WIDGLAG2 data set specifies a Start-to-Start lag of nine days between the activity 'Prototype' and its successors, 'Materials' and 'Facility,' and a Finish-to-Start lag of two days between 'Facility' and 'Init. Prod.'. In addition, changes to the widget design are permitted to be made no earlier than six days after in-house evaluation of the product has begun. Furthermore, the Engineering department has to ensure that there will be at least three days available for any changes that need to be carried out after the test market results have come in. These constraints are incorporated in the WIDGLAG data set by setting the value of the LAGDUR variable equal to 'ss_6' for the relationship between 'Evaluate' and 'Changes' and equal to 'ff_3' for the relationship between 'Test Market' and 'Changes.'

The project is scheduled using PROC CPM subject to weekends and the holidays defined in the HOLIDAYS data set. Specifying the COLLAPSE option in the PROC CPM statement ensures that there is one observation per activity. The WIDGLAGH data set is created by deleting the successor variable from the Schedule data set produced by PROC CPM.

Since there is no precedence information contained in the WIDGLAGH data set, specifying DATA=WIDGLAGH in the PROC GANTT statement without the PRECDATA= option produces a nonprecedence Gantt chart. You can produce a Logic Gantt chart by specifying the precedence information using the PRECDATA= option in the PROC GANTT statement as long as the activity variable is common to both the schedule and Precedence data sets.

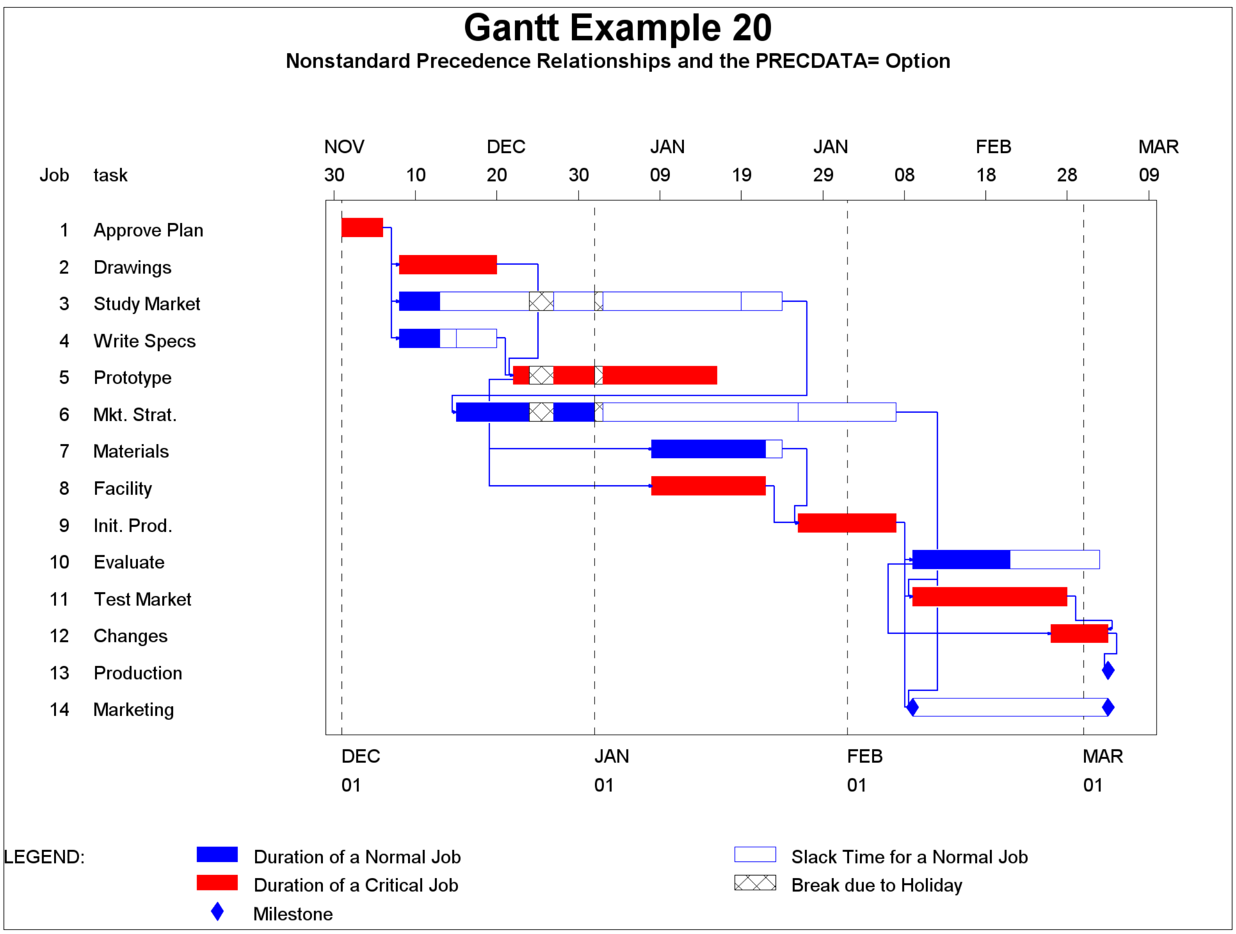
The Gantt chart shown in Output 8.20.1 is produced by specifying PRECDATA= WIDGLAG2. The lag type of the precedence connections is indicated to PROC GANTT using the LAG= option in the CHART statement. The width of the precedence connections is set to 2 with the WPREC= option, and the color of the connections is set to blue using the CPREC= option. The MININTGV= and MINOFFLV= options are specified in the CHART statement in an attempt to minimize the number of 5-segment connections. A reference line with a line style of 2 is drawn at the beginning of every month by using the REF= and LREF= options in the CHART statement.

```
options ps=60 ls=100;

title h=2 'Gantt Example 20';

 /* Activity-on-Node representation of the project with lags */
data widglag2;
   format task $12. succ $12. lagdur $4. ;
   input task & days succ & lagdur $ ;
   datalines;
Approve Plan   5  Drawings      .
```

```
Approve Plan   5  Study Market  .
Approve Plan   5  Write Specs   .
Drawings      10  Prototype     .
Study Market   5  Mkt. Strat.   .
Write Specs    5  Prototype     .
Prototype     15  Materials     ss_9
Prototype     15  Facility      ss_9
Mkt. Strat.   10  Test Market   .
Mkt. Strat.   10  Marketing     .
Materials     10  Init. Prod.   .
Facility      10  Init. Prod.   fs_2
Init. Prod.   10  Test Market   .
Init. Prod.   10  Marketing     .
Init. Prod.   10  Evaluate      .
Evaluate      10  Changes       ss_6
Test Market   15  Changes       ff_3
Changes        5  Production    .
Production     0  .             .
Marketing      0  .             .
;

data holidays;
   format holiday holifin date7.;
   input holiday & date7. holifin & date7. holidur;
   datalines;
24dec03  26dec03  4
01jan04  .        .
;

proc cpm data=widglag2 holidata=holidays date='1dec03'd
         interval=weekday collapse;
   activity task;
   succ     succ / lag = (lagdur);
   duration days;
   holiday  holiday / holifin=(holifin);
   run;

data widglagh;
   set _last_;
   drop succ;
   run;

* set up required pattern statements;
pattern1 c=blue  v=s;  /* duration of a noncrit. activity  */
pattern2 c=blue  v=e;  /* slack time for a noncrit. act.   */
pattern3 c=red   v=s;  /* duration of a critical activity  */
pattern4 c=red   v=e;  /* slack time for a supercrit. act. */
pattern5 c=red   v=r2; /* duration of a supercrit. act.    */
pattern6 c=cyan  v=s;  /* actual duration of an activity   */
pattern7 c=black v=x1; /* break due to a holiday           */

* set graphics options;
goptions vpos=50 hpos=100 htext=1.025;
```

```
title2 c=black h=1.5
   'Nonstandard Precedence Relationships and the PRECDATA= Option';

proc gantt graphics data=widglagh precdata=widglag2
                    holidata=holidays;
   chart / compress dur=days height=1.5
           holiday=(holiday) holifin=(holifin)
           cmile=blue cprec=blue wprec=2
           ref='01dec03'd to '01mar04'd by month
           cref=black lref=2 reflabel caxis=black
           act=task succ=(succ) lag=(lagdur)
           minintgv=2 minofflv=.5;
   id task;
   run;
```

Output 8.20.1 Nonstandard Precedence Relationships



Example 8.21: Using the SAS/GRAPH ANNOTATE= Option

This example illustrates the use of the ANNOTATE= option to add graphics and text to the body of the Gantt chart. The intent of the first invocation of PROC GANTT is to display the resource requirements of each activity on the Gantt chart, while that of the second invocation is to plot the resource usage bar chart for the replenishable resource engineers and the resource availability curve for the consumable resource cost.

The data for this example come from Example 4.15, in which the widget manufacturing project is scheduled using PROC CPM subject to resource constraints. The project network is defined in the WIDGRES data set using AOA format. The number of engineers needed per day per activity is a replenishable resource and is identified by the ENGINEER variable in the WIDGRES data set. The cost incurred per day per activity is a consumable resource and is identified by the ENGCOST variable in the WIDGRES data set. The WIDGRIN data set specifies the resource availabilities for the project. The schedule produced by PROC CPM using the default choice of LST as a heuristic is shown in Output 8.21.1. The following programs assume that the schedule is stored in the WIDGSCH2 data set and that the resource usage is stored in the WIDGROU2 data set.

The Annotate macros are used in this example to simplify the process of creating Annotate observations. The ANNOMAC macro is first used to compile the Annotate macros and make them available for use. The Annotate data set ANNO1 is then created using the Annotate macros. The DCLANNO macro declares all Annotate variables except the TEXT variable, and the SYSTEM macro defines the Annotate reference system. The coordinate system defined here uses *date* for the horizontal scale and *job number* for the vertical scale. The text to be displayed contains the number of engineers required per day and the total cost over the duration of the activity. The LABEL macro is used to annotate the necessary text on the Gantt chart using the BRUSH font.

The GANTT procedure is invoked with the ANNOTATE=ANNO1 specification in the PROC GANTT statement. The resulting Gantt chart is shown in Output 8.21.2. It is important to note that the job number will be used for the vertical scale even if NOJOBNUM is specified in the CHART statement.

Output 8.21.1 Resource Constrained Schedule: Rule = LST

Resource Constrained Schedule: Rule = LST

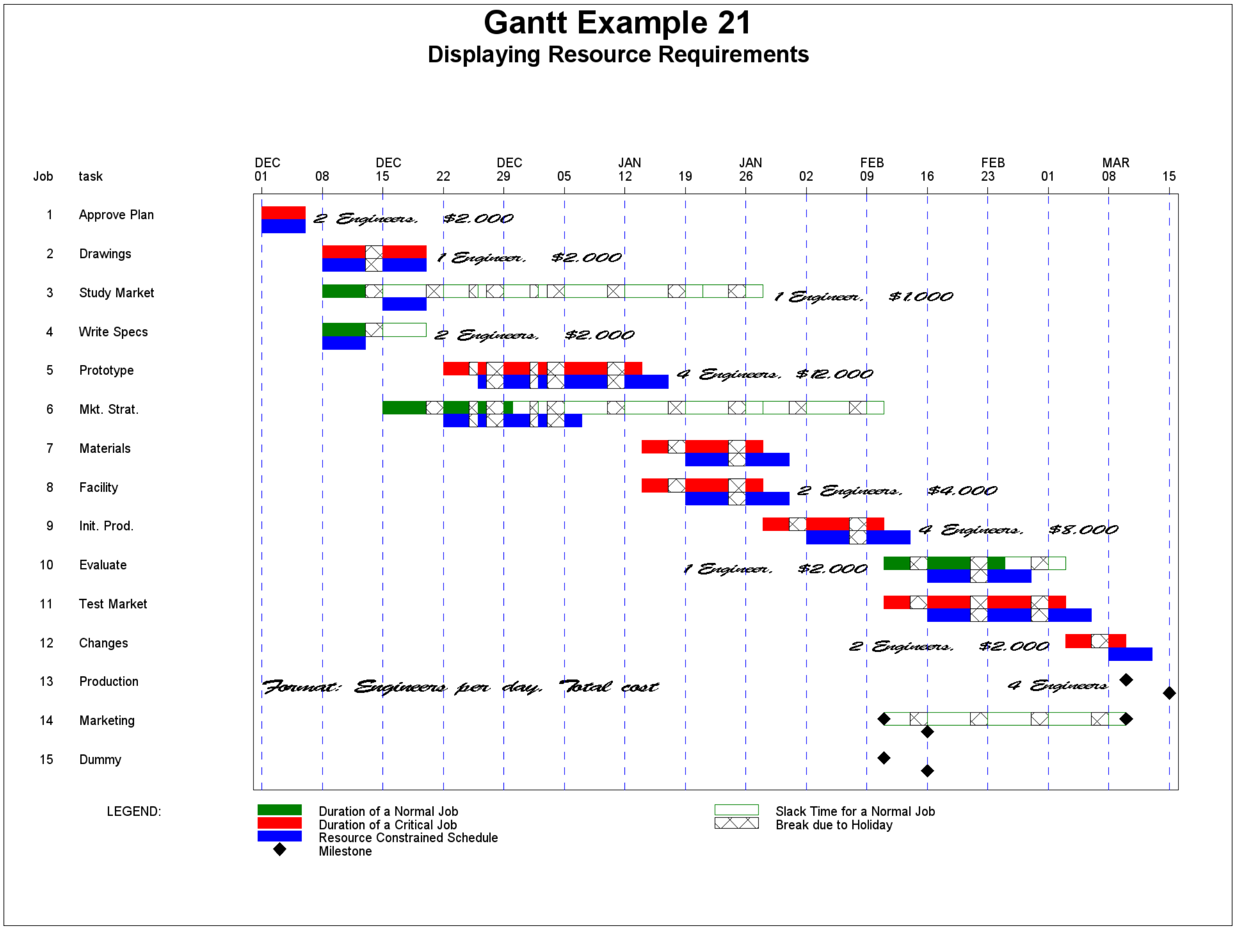
Obs	tail	head	days	task	engineer	engcost	S_START	S_FINISH	E_START	E_FINISH
1	1	2	5	Approve Plan	2	400	01DEC03	05DEC03	01DEC03	05DEC03
2	2	3	10	Drawings	1	200	08DEC03	19DEC03	08DEC03	19DEC03
3	2	4	5	Study Market	1	200	15DEC03	19DEC03	08DEC03	12DEC03
4	2	3	5	Write Specs	2	400	08DEC03	12DEC03	08DEC03	12DEC03
5	3	5	15	Prototype	4	800	26DEC03	16JAN04	22DEC03	13JAN04
6	4	6	10	Mkt. Strat.	.	.	22DEC03	06JAN04	15DEC03	29DEC03
7	5	7	10	Materials	.	.	19JAN04	30JAN04	14JAN04	27JAN04
8	5	7	10	Facility	2	400	19JAN04	30JAN04	14JAN04	27JAN04
9	7	8	10	Init. Prod.	4	800	02FEB04	13FEB04	28JAN04	10FEB04
10	8	9	10	Evaluate	1	200	16FEB04	27FEB04	11FEB04	24FEB04
11	6	9	15	Test Market	.	.	16FEB04	05MAR04	11FEB04	02MAR04
12	9	10	5	Changes	2	400	08MAR04	12MAR04	03MAR04	09MAR04
13	10	11	0	Production	4	800	15MAR04	15MAR04	10MAR04	10MAR04
14	6	12	0	Marketing	.	.	16FEB04	16FEB04	11FEB04	11FEB04
15	8	6	0	Dummy	.	.	16FEB04	16FEB04	11FEB04	11FEB04

Obs	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	01DEC03	05DEC03	0		
2	08DEC03	19DEC03	0		
3	21JAN04	27JAN04	5	engineer	
4	15DEC03	19DEC03	0		
5	22DEC03	13JAN04	3	engineer	
6	28JAN04	10FEB04	0		
7	14JAN04	27JAN04	0		
8	14JAN04	27JAN04	0		
9	28JAN04	10FEB04	0		
10	18FEB04	02MAR04	0		
11	11FEB04	02MAR04	0		
12	03MAR04	09MAR04	0		
13	10MAR04	10MAR04	0		
14	10MAR04	10MAR04	0		
15	11FEB04	11FEB04	0		

```
title c=black h=1.75 'Gantt Example 21';
title2 c=black h=1.25 'Displaying Resource Requirements';

* set background to white and text to black;
goptions ctext=black cback=white;

* set graphics options;
goptions vpos=50 hpos=100 htext=1.01;

* begin annotate process;

* compile annotate macros;
%annomac;
```

```
* create annotate data set for first chart;
data anno1;
   %dclanno;       /* set length and type for annotate variables */
   %system(2,2,4); /* define annotate reference system           */
   set widgsch2;
   length lab $20;
   length TEXT $ 37;
   Y1 = _n_;
   lab='    ';

   if _n_=1 then do;
      %label('01dec03'd,13,
             'Format: Engineers per day, Total cost',*,0,0,1.2,brush,6);
      end;

   if engineer ^= . then do;
      /* create a text label */
      lab = put(engineer, 1.) || " Engineer";
      if engineer > 1 then lab = trim(lab) || "s";
      if days > 0 then lab = trim(lab) || ", " ||
                                             put(engcost*days, dollar7.);

      /* position the text label */
      if y1 < 10 then do;
         x1 = max(l_finish, s_finish) + 2;
         %label(x1,y1,lab,black,0,0,1.0,brush, 6);
         end;
      else do;
         x1 = e_start - 2;
         %label(x1,y1,lab,black,0,0,1.0,brush, 4);
         end;
      end;
run;

* annotate the Gantt chart;
proc gantt graphics data=widgsch2 holidata=holdata annotate=anno1;
chart / pcompress holiday=(hol) interval=weekday increment=7
        ref='1dec03'd to '21mar04'd by week
        cref=blue lref=2
        dur=days cmile=black caxis=black;
id task;
run;
```

Output 8.21.2 Using the ANNOTATE= Option



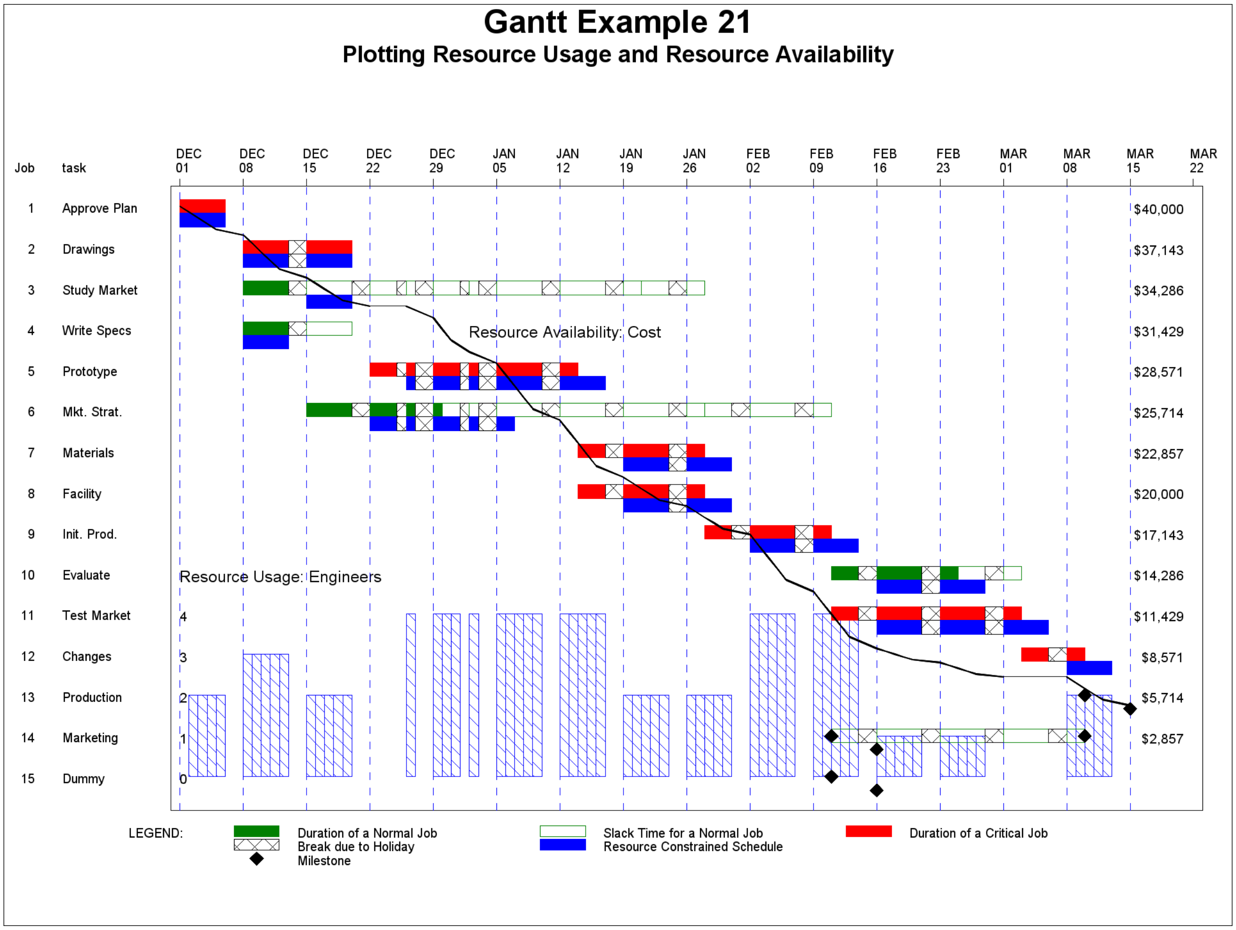
The next illustration of the ANNOTATE= option is to plot the resource usage bar chart for the replenishable resource engineers and the resource availability curve for the consumable resource cost. A DATA step determines the largest value of the cost availability throughout the life of the project in order to scale the costs accordingly. The CSCALE macro variable is required to represent cost availabilities on the Gantt chart. Since there are no further cash inflows after December 1, 2003, and there are 15 jobs represented on the chart, the value of the macro variable CSCALE is $(15 - 1)/40000$.

An Annotate data set, ANNO2, is created in much the same fashion as ANNO1, but it employs some additional macros. The BAR macro is used to draw the resource usage bar chart, and the DRAW and MOVE macros are used to draw the resource availability curve. The PUSH and POP macros are used as necessary to store and retrieve the last used coordinates from the stack, respectively. The resulting Gantt chart is displayed in Output 8.21.3.

```
* calculate scaling factor for cost curve;
data _null_;
   set widgrou2 end=final;
   retain maxcost;
   if aengcost > maxcost then maxcost=aengcost;
   if final then call symput('cscale',14/maxcost);
   run;

* create annotate data set for second chart;
data anno2;
   %dclanno;        /* set length and type for annotate variables */
```

```
   %system(2,2,4); /* define annotate reference system           */
   set widgrou2;
   length lab $16;
   length TEXT $27;
   x1=_time_;
   y1=15-aengcost*symget('cscale');
   y2=15-rengineer;
   lab='    ';

   if _n_=1 then do;
      /* print labels */
      do i = 1 to 14 by 1;
         lab=put( (15-i) / symget('cscale'), dollar7.);
         %label('21mar04'd,i,lab,black,0,0,1.0,,4);
         end;
      do i = 0 to 4 by 1;
         lab=put(i,1. );
         %label('01dec03'd,15-i,lab,black,0,0,1.0,,6);
         end;
      %label('01dec03'd,10,
             'Resource Usage: Engineers',*,0,0,1.2,,6);
      %label('02jan04'd,4,
             'Resource Availability: Cost',*,0,0,1.2,,6);
      %move(x1,y1);
      %push;
      end;
   else do;
      /* draw cost availability curve */
      %pop;
      when='a';
      %draw(x1,y1,black,1,2);
      %push;
      /* draw engineer usage barchart */
      when='b';
      if y2 <= 14 then do;
         %bar(x1,15,x1+1,y2,blue,0,l1);
         end;
      end;
   run;

title c=black h=1.75 'Gantt Example 21';
title2 c=black h=1.25
       'Plotting Resource Usage and Resource Availability';

* annotate the Gantt chart;
proc gantt graphics data=widgsch2 holidata=holdata annotate=anno2;
chart / pcompress holiday=(hol) interval=weekday increment=7
        mindate='1dec03'd maxdate='21mar04'd
        ref='1dec03'd to '21mar04'd by week
        cref=blue lref=2
        dur=days cmile=black caxis=black;
id task;
run;
```

Output 8.21.3 Using the ANNOTATE= Option



Example 8.22: Using the Automatic Text Annotation Feature

The following example is a subproject of a larger project involving the maintenance of a pipeline and steam calender (Moder, Phillips, and Davis 1983), and it illustrates the automatic text annotation feature of the GANTT procedure. The SHUTDOWN data set is input as the activity data set to PROC CPM, and the project is scheduled to begin on June 1, 2004. PROC GANTT is used to produce a Gantt chart of the resulting schedule with the data set LABELS specified as a Label data set; the output is shown in Output 8.22.1. The LABVAR= option in the CHART statement specifies the ACT variable as the common linking variable. The LABSPLIT= option is specified in order to prevent the labels from splitting on embedded blanks.

The first observation in the LABELS data set causes the value of the ACT variable to be displayed at the E_START time for every activity in the project. The value of _YOFFSET='-.2' positions the baseline of the displayed text at 0.2 barheights above the top of the first bar for the activity. Similarly the second observation displays the ID variable at the E_START time for each activity with the baseline positioned at 0.8 barheights below the bottom of the first bar for the activity. The heights for both these strings is 1 barheight. The next two observations in the LABELS data set display the symbols corresponding to the values 'N' and 'M' in the ORFONT font, rotated at an angle of 90 degrees, beside the milestones corresponding to the deactivation and activation of the calender, respectively. Observations 5 and 6 indicate the start and finish of the "Maintenance Period" by displaying the indicated strings rotated 90 degrees at the start and finish times of the activity

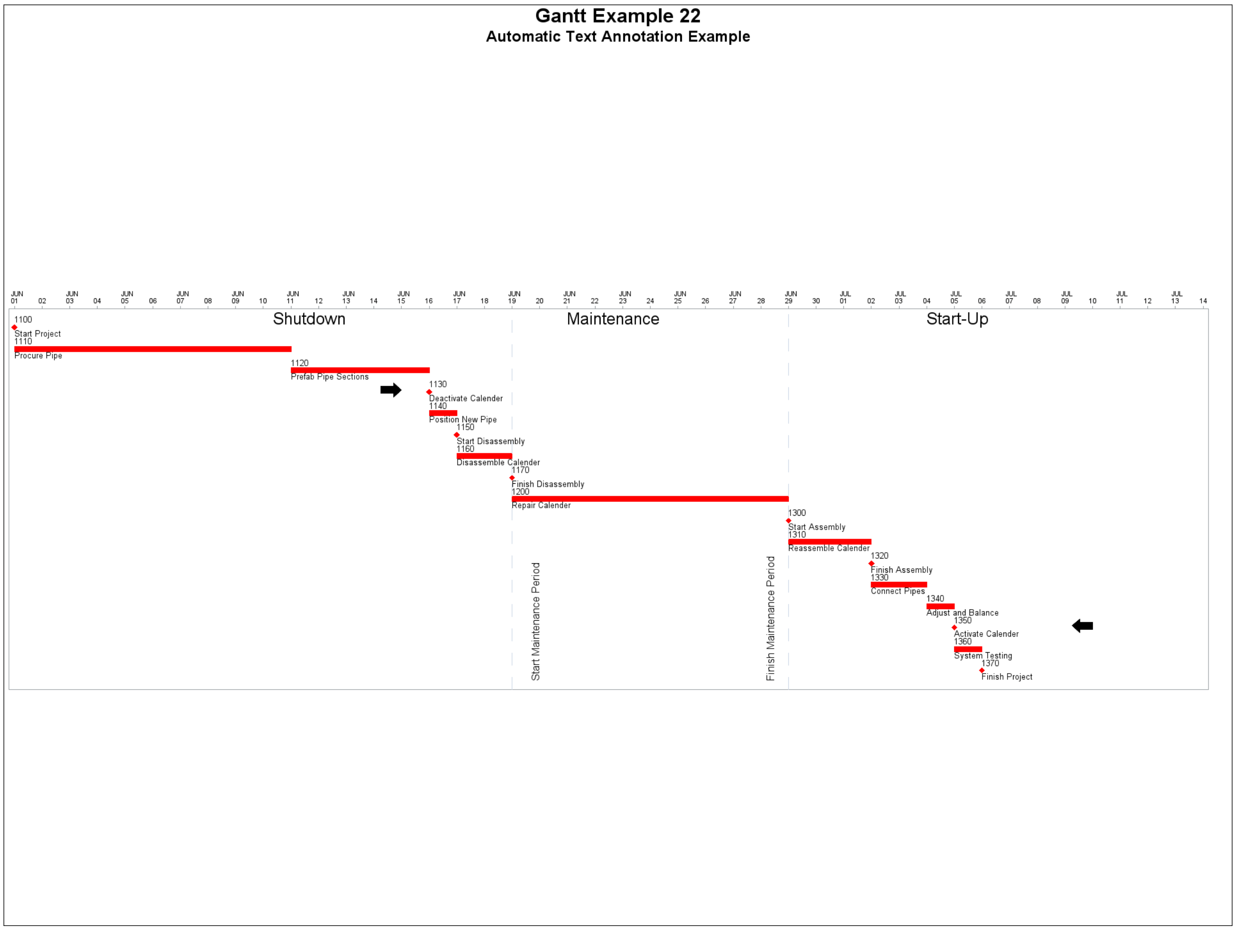
'Repair Calender.' Finally, the last three observations provide headings for each of the three distinct regions on the chart. The _JLABEL variable is used along with the _XVAR variable to place the strings in the regions defined by the start and finish times of the 'Repair Calender' activity.

It should be noted that since the plot times are linked to variables rather than absolute values, the Label data set need not be changed even if the project is rescheduled. This is a convenient feature when monitoring a project in progress, since the annotation automatically places the labels at the appropriate times.

```
title c=black 'Gantt Example 22';

data shutdown;
  input act succ id & $20. dur;
  datalines;
1100 1110 Start Project         0
1110 1120 Procure Pipe         10
1120 1130 Prefab Pipe Sections  5
1130 1140 Deactivate Calender   0
1140 1150 Position New Pipe     1
1150 1160 Start Disassembly     0
1160 1170 Disassemble Calender  2
1170 1200 Finish Disassembly    0
1200 1300 Repair Calender      10
1300 1310 Start Assembly        0
1310 1320 Reassemble Calender   3
1320 1330 Finish Assembly       0
1330 1340 Connect Pipes         2
1340 1350 Adjust and Balance    1
1350 1360 Activate Calender     0
1360 1370 System Testing        1
1370 .    Finish Project        0
;

proc cpm data=shutdown date='01jun04'd interval=day
  out=sched;
  act act;
  succ succ;
  dur dur;
  id id;
run;

data labels;
  input act _y _xvar $ _lvar $ _yoffset _xoffset _label & $25.
        _alabel _hlabel _jlabel $ _flabel $ _clabel $;
  datalines;
.    -1 e_start  act -.3  0 .                           0 1.5 .   .      .
.    -1 e_start  id  2.3  0 .                           0 1.5 .   .      .
1130  . e_start  .   1.5 -1 N                          90   2 L orfont   .
1350  . e_finish .   1.5  5 M                          90   2 L orfont   .
1200 17 e_start  .   2.5  1 Start Maintenance Period   90   2 .   .      .
1200 17 e_finish .   2.5 .5 Finish Maintenance Period  90   2 .   .      .
1200  1 e_start  .     . -6 Shutdown                    0   3 R   .      .
1200  1 e_start  .     .  2 Maintenance                 0   3 L   .      .
1200  1 e_finish .     .  6 Start-Up                    0   3 L   .      .
;
```

```
title2 'Automatic Text Annotation Example';

proc gantt data=sched labdata=labels graphics maxdec=0;
  chart / pcompress nolegend nojobnum dur=dur
          mininterval=day scale=5 skip=3 maxdate='14jul04'd
          labvar=act labsplit='/' ref='19jun04'd '29jun04'd
          height=1.5 lref=20;
run;
```

Output 8.22.1 Using the LABDATA= Option



Example 8.23: Multiproject Gantt Charts

The following example illustrates an application of the PATTERN variable to display summary bars for subprojects. The LAN Selection Project (Bostwick 1986) consists of eight subprojects, two of which represent the beginning and ending of the master project. The data set LANACT defines the structure of the project. The ACT and SUCC variables define the precedence relationships, the PARENT variable defines the parent task, and the DAYS variable contains the duration of the activity.

The project is scheduled using the CPM procedure with a PARENT statement to identify the parent. The schedule data set, SCHED, is created by appending a _PATTERN variable to the output data set generated by CPM. The value of this variable is set to '4,' corresponding to subprojects, and set to missing otherwise. This results in the subproject bars being filled using PATTERN4, namely a solid black pattern. The ACTID variable is indented within the DATA step to reflect the level of each activity in the project hierarchy when used as the ID variable.

A Label data set, LABELS, is created in order to add markers to both ends of the schedule bars that correspond to subprojects. The two observations in the LABELS data set are linked to the SCHED data set with the _PATTERN variable.

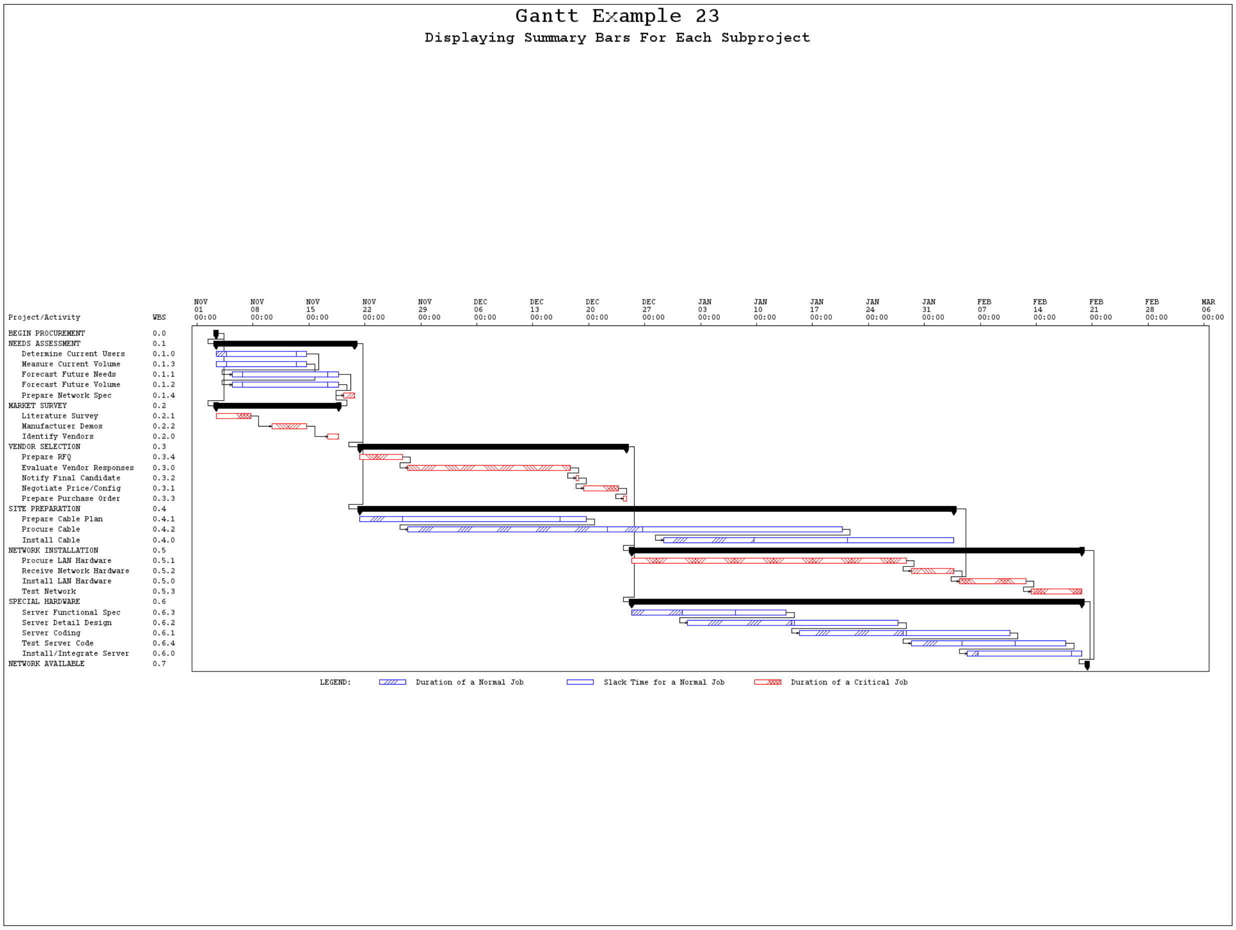
The GANTT procedure is next invoked to produce the Gantt chart in Output 8.23.1. The LABVAR=_PATTERN specification establishes the link between the Schedule and Label data sets. The ACT= and SUCC= options are used to display the precedence relationships between activities.

```
pattern1 c=blue   v=r5;         /* Non-critical duration  */
pattern2 c=blue   v=e;          /* Slack duration         */
pattern3 c=red    v=x5;         /* Critical duration      */
pattern4 c=black  v=s;          /* Project duration       */

data lanact;
   format act $30. succ $30. parent $20.;
   input act & succ & parent & days;
   datalines;
Measure Current Volume     Forecast Future Volume     NEEDS ASSESSMENT       2
Literature Survey          Manufacturer Demos         MARKET SURVEY          5
Determine Current Users    Forecast Future Needs      NEEDS ASSESSMENT       2
Forecast Future Volume     Prepare Network Spec       NEEDS ASSESSMENT       2
Manufacturer Demos         Identify Vendors           MARKET SURVEY          5
Forecast Future Needs      Prepare Network Spec       NEEDS ASSESSMENT       2
Identify Vendors           .                          MARKET SURVEY          2
Prepare Network Spec       .                          NEEDS ASSESSMENT       2
Prepare RFQ                Evaluate Vendor Responses  VENDOR SELECTION       4
Prepare Cable Plan         Procure Cable              SITE PREPARATION       4
Evaluate Vendor Responses  Notify Final Candidate     VENDOR SELECTION      15
Procure Cable              Install Cable              SITE PREPARATION      22
Notify Final Candidate     Negotiate Price/Config     VENDOR SELECTION       1
Install Cable              .                          SITE PREPARATION      10
Negotiate Price/Config     Prepare Purchase Order     VENDOR SELECTION       3
Prepare Purchase Order     .                          VENDOR SELECTION       1
Server Functional Spec     Server Detail Design       SPECIAL HARDWARE       5
Procure LAN Hardware       Receive Network Hardware   NETWORK INSTALLATION  25
Server Detail Design       Server Coding              SPECIAL HARDWARE      10
Receive Network Hardware   Install LAN Hardware       NETWORK INSTALLATION   4
Server Coding              Test Server Code           SPECIAL HARDWARE      10
Install LAN Hardware       Test Network               NETWORK INSTALLATION   7
Test Server Code           Install/Integrate Server   SPECIAL HARDWARE       5
Test Network               .                          NETWORK INSTALLATION   5
Install/Integrate Server   .                          SPECIAL HARDWARE       2
BEGIN PROCUREMENT          NEEDS ASSESSMENT           .                      .
BEGIN PROCUREMENT          MARKET SURVEY              .                      .
NEEDS ASSESSMENT           VENDOR SELECTION           .                      .
NEEDS ASSESSMENT           SITE PREPARATION           .                      .
MARKET SURVEY              Prepare Network Spec       .                      .
```

```
VENDOR SELECTION           NETWORK INSTALLATION       .                      .
VENDOR SELECTION           SPECIAL HARDWARE           .                      .
SITE PREPARATION           Install LAN Hardware       .                      .
NETWORK INSTALLATION       NETWORK AVAILABLE          .                      .
SPECIAL HARDWARE           NETWORK AVAILABLE          .                      .
;

proc sort data=lanact;
   by act;
   run;

proc cpm data=lanact out=lanout
     expand interval=workday date='03nov03'd;
   parent parent / wbs eso;
   activity act;
   duration days;
   successor succ;
   run;

 /* create the schedule data set with a pattern variable */
data sched;
   label wbs_code='WBS';
   label actid='Project/Activity';
   set lanout;
   if proj_lev !0 then do;
   if parent='' then _pattern=4;
      actid=act;
      do i=1 to proj_lev-1;
         actid = "   " || actid;
      end;
      output;
   end;
;

proc sort data=sched;
   by es_asc wbs_code;
   run;

 /* create the label data set */
data labels;
   _pattern=4;
   _flabel='orfont';
   _jlabel='c';
   _yoffset=0.925;
   _label='Z';
   _xvar='e_start ';
   output;
   _xvar='l_finish';
   output;
;

title1 f='Cumberland AMT' h=1.75 'Gantt Example 23';
title2 f='Cumberland AMT' h=1.25 'Displaying Summary Bars For Each Subproject';
```

```
proc gantt graphics data=sched labdata=labels;
  id actid wbs_code;
  chart / pcompress nojobnum ctext=black caxis=black
          mindate='01nov03'd maxdate='29feb04'd increment=7
          labvar=_pattern font='Cumberland AMT' height=1.5
          minoffgv=1.5 minofflv=1.5 cprec=black wprec=1
          scale=1.5 act=act succ=succ;
run;
```

Output 8.23.1 Using the PATTERN Variable and Labels



Example 8.24: Multisegment Gantt Charts

The following is a simple example that illustrates the generation of multisegmented Gantt charts. The SCHED data set identifies the city, the arrival time, and the departure time for each of four traveling salespeople. In addition, a _PATTERN variable is used to identify the pattern to be used for drawing the bar. The objective is to display the complete schedule for each sales person on a single row. This would require displaying several bars on a single row, each bar corresponding to the time spent in a city. In order to do this, you need first to sort the SCHED data set by Salesperson and Arrival Time and then to add a SEGMT_NO variable that identifies the number of the segment that, in this case, is the order in which the salesperson visits the city. The resulting data set, NEWSCHED, is shown in Output 8.24.1. You next create the LABELS data set in order to identify the names of the cities above the bars; the resulting Gantt chart is shown in Output 8.24.2.

Notice that each bar is drawn using the pattern identified by the _PATTERN variable in the SCHED data set. In the absence of the _PATTERN variable, the pattern associated with the resource-constrained schedule would have been used for all the bars. This is the same mechanism that produced the split segments in Example 8.13 although the SEGMT_NO variable in this case was automatically created by the CPM procedure.

```
data sched;
   format city $12. from to date7. ;
   input person $ city & from & date7. to & date7. _pattern;
   datalines;
Clark   New York     01May04  03May04  10
Clark   Boston       06May04  09May04  11
Clark   Wisconsin    12May04  15May04  12
Clark   Chicago      18May04  24May04  13
Clark   New York     28May04  02Jun04  10
Stevens Charlotte    02May04  04May04  14
Stevens Atlanta      08May04  10May04  15
Stevens Dallas       12May04  15May04  16
Stevens Denver       17May04  20May04  17
Stevens Nashville    27May04  02Jun04  18
Stevens Charlotte    04Jun04  06Jun04  14
Jackson Los Angeles  01May04  08May04  19
Jackson Las Vegas    11May04  18May04  20
Jackson Portland     21May04  23May04  21
Jackson Seattle      25May04  29May04  22
Rogers  Miami        02May04  07May04  23
Rogers  Tampa        11May04  15May04  24
Rogers  New Orleans  18May04  24May04  25
Rogers  Houston      28May04  01Jun04  26
;

 /* Sort data by person, from */
proc sort data=sched;
   by person from;
   run;

/* Add Segmt_no variable */
data newsched;
   set sched;
   retain segmt_no;
   if person ne lag(person) then segmt_no=1;
   else segmt_no = segmt_no + 1;
   output;
   run;

proc print data=newsched;
   title2 'Data NEWSCHED';
   run;

data labels;
   _y=-1;
   _lvar="city";
   _xvar="from";
   _flabel="";
   _hlabel=0.75;
```

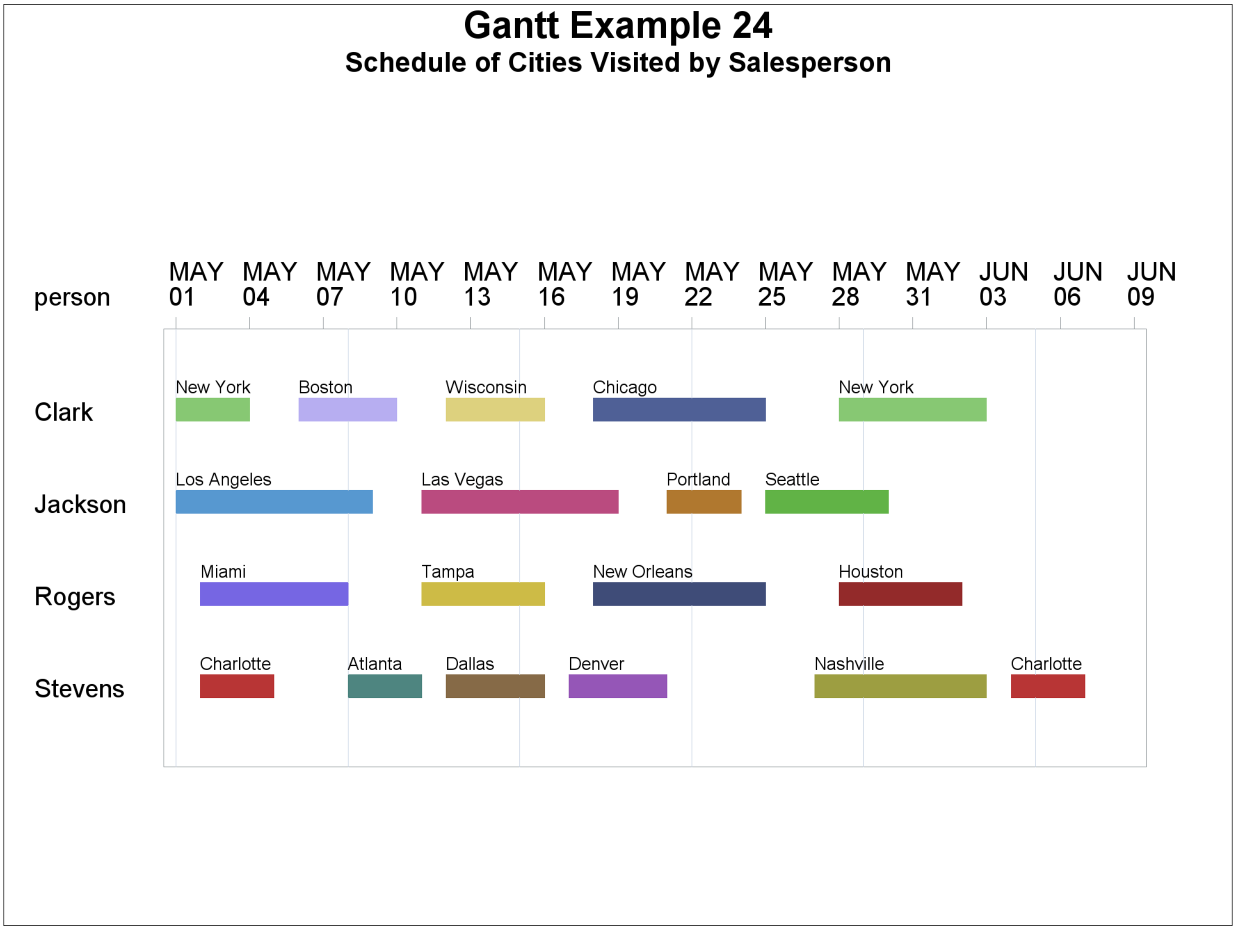
```
   _yoffset = -.2;
   run;

* set up required pattern statements;
pattern1 v=s r=25;

* set graphics options;
proc gantt graphics data=newsched labdata=labels;
id person;
chart / ss=from sf=to compress labsplit='.' scale=2
        nolegend nojobnum skip=3
        ref='01may04'd to '30jun04'd by week;
run;
```

Output 8.24.1 NEWSCHED Data Set

Data NEWSCHED

Obs	city	from	to	person	_pattern	segmt_no
1	New York	01MAY04	03MAY04	Clark	10	1
2	Boston	06MAY04	09MAY04	Clark	11	2
3	Wisconsin	12MAY04	15MAY04	Clark	12	3
4	Chicago	18MAY04	24MAY04	Clark	13	4
5	New York	28MAY04	02JUN04	Clark	10	5
6	Los Angeles	01MAY04	08MAY04	Jackson	19	1
7	Las Vegas	11MAY04	18MAY04	Jackson	20	2
8	Portland	21MAY04	23MAY04	Jackson	21	3
9	Seattle	25MAY04	29MAY04	Jackson	22	4
10	Miami	02MAY04	07MAY04	Rogers	23	1
11	Tampa	11MAY04	15MAY04	Rogers	24	2
12	New Orleans	18MAY04	24MAY04	Rogers	25	3
13	Houston	28MAY04	01JUN04	Rogers	26	4
14	Charlotte	02MAY04	04MAY04	Stevens	14	1
15	Atlanta	08MAY04	10MAY04	Stevens	15	2
16	Dallas	12MAY04	15MAY04	Stevens	16	3
17	Denver	17MAY04	20MAY04	Stevens	17	4
18	Nashville	27MAY04	02JUN04	Stevens	18	5
19	Charlotte	04JUN04	06JUN04	Stevens	14	6

Output 8.24.2 Multisegment Gantt Chart

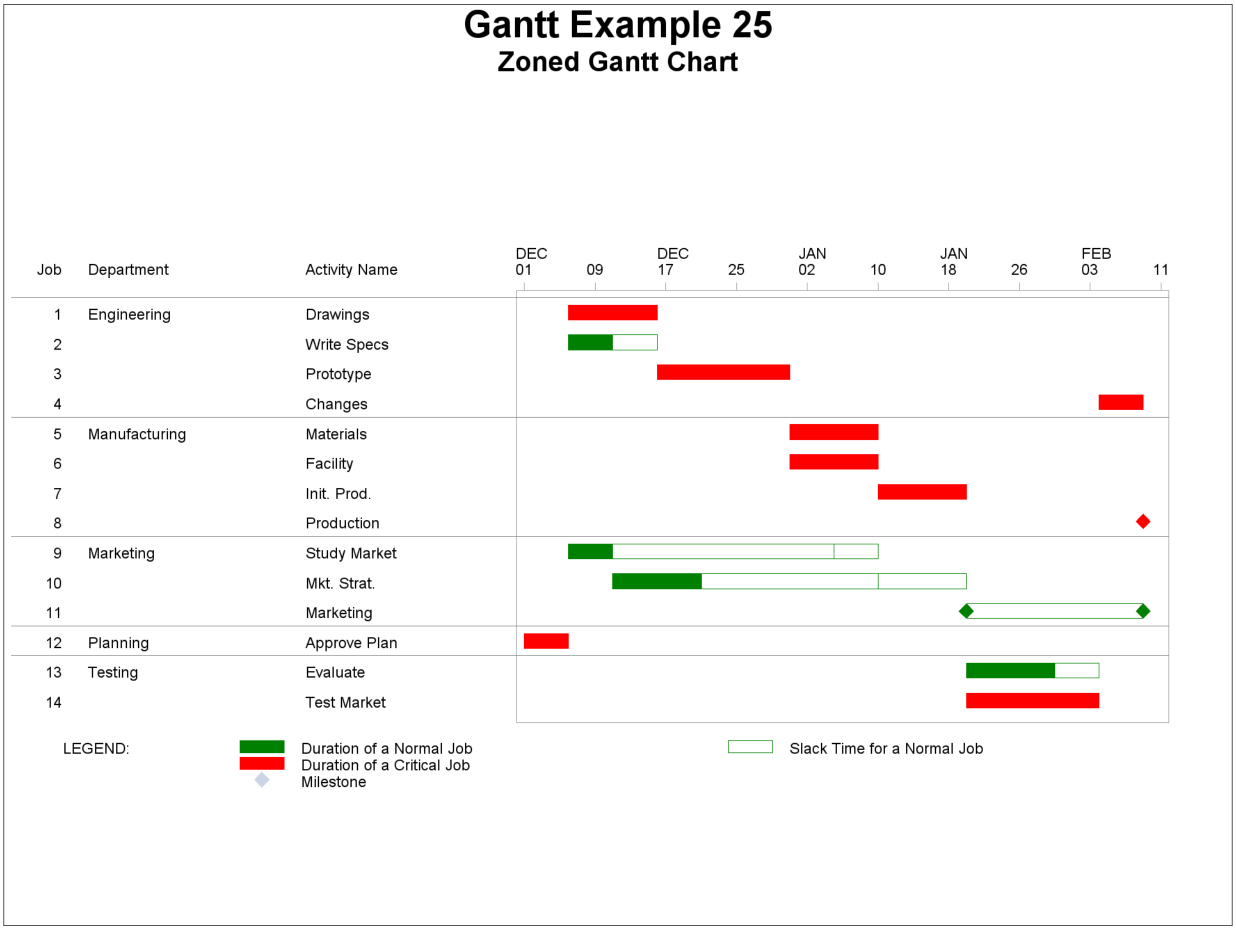


Example 8.25: Zoned Gantt Charts

Example 8.15 illustrated the use of BY processing with the GANTT procedure to present separate Gantt charts for each department. Alternatively, you can use a zoned Gantt chart to display each of the departmental schedules on the same chart with the different department schedules separated by horizontal zone lines running across the chart. The ZONE variable divides the Activity axis into distinct zones. Activities with the same value of the ZONE variable belong to the same zone. This example produces a zoned Gantt chart using the schedule data from Example 8.15. The ZONE=DEPT specification in the CHART statement identifies the DEPT variable as the ZONE variable. The ONEZONEVAL option specifies that the value of the ZONE variable be displayed only when beginning new zones. The resulting Gantt chart is shown in Output 8.25.1. You can customize the color, style and width of the zone line by using the CZONE=, LZONE=, and WZONE= options, respectively. You can also control the span and offset of the zone line by specifying the ZONESPAN= and ZONEOFF= options, respectively, in the CHART statement.

```
proc cpm date='01dec03'd data=widgetn;
   activity task;
   duration days;
   successor succ1 succ2 succ3;
   id dept;
   run;

proc sort;
   by dept e_start;
   run;

proc gantt split='/';
   chart / pcompress scale=1 dur=days
           mindate='01dec03'd maxdate='11feb04'd
           zone=dept onezoneval czone=gray;
   id task;
   run;
```

Output 8.25.1 Gantt Charts Zoned by Department



Example 8.26: Web-Enabled Gantt Charts

This example illustrates the process of “Web-enabling” your Gantt charts. This feature enables you to associate a URL with each activity on a Gantt chart. By using this feature together with SAS/IntrNet software, you can develop some very powerful Project Management applications. SAS/IntrNet software provides you with the capability to perform data set queries and execute SAS applications in real time and view the results in HTML format using a Web browser.

This example takes advantage of the Output Delivery System (ODS) HTML statement to create a very simple “drill-down” Gantt application beginning from a summary Gantt chart of the “top level” projects in Example 8.23. The objective is to display a detailed Gantt chart of the activities in a subproject when you click on the subproject bar.

In order to be able to click on an activity and invoke an action, you need to add variables to the schedule data set that associate a URL with each of the activities that you want linked. The following code adds the WEBVAR and WEBVAR2 variables to the LANOUT data set in Example 8.23 to create the LANWEB data set. The WEBVAR variable uses the ALT= portion to identify information about an activity’s schedule that is to be displayed when the mouse hovers over the schedule bar. In addition, it uses the HREF= portion to associate the URL with the linked activity. The WEBVAR2 variable uses only the ALT= portion, so information in the detailed Gantt chart can still be displayed by hovering over the schedule bars.

The LANWEB data set is then sorted by the WBS_CODE variable.

```
data lanweb;
   set lanout;
   length webvar $500;
   length webvar2 $500;

   /* WEBVAR is for the top-level summary chart */
   webvar='alt='|| quote(
     'Activity: '||trim(left(act))||'0D'x||
     '------------------------'||'0D'x||
     'Early Start: '||put(e_start,  datetime.)||'0D'x||
     'Early Finish:'||put(e_finish, datetime.)||'0D'x||
     'Late Start:  '||put(l_start,  datetime.)||'0D'x||
     'Late Finish: '||put(l_finish, datetime.) )||
     ' HREF=#'||trim(wbs_code)  /* link to the anchors */
     ;

   /* WEBVAR2 is for the detailed charts */
   webvar2='alt='|| quote(
     'Activity: '||trim(left(act))||'0D'x||
     '------------------------'||'0D'x||
     'Early Start: '||put(e_start,  datetime.)||'0D'x||
     'Early Finish:'||put(e_finish, datetime.)||'0D'x||
     'Late Start:  '||put(l_start,  datetime.)||'0D'x||
     'Late Finish: '||put(l_finish, datetime.) )
     ;
   run;

proc sort data=lanweb;
   by wbs_code;
   run;
```

Before creating the charts, you need to specify that the GIF driver be used to create graphics output. ODS HTML output always creates a "body" file, which is a single HTML document containing the output from one or more procedures and is specified using the FILE= option in the ODS HTML statement.

```
goptions reset=all device=gif;

ods html file="Gantt_Sum.html";
```

For example, when you click on any of the schedule bars for an activity with WBS_CODE='0.2', you link to an anchor labeled '0.2' in the body file Gantt_Sum.html.

You are now ready to create the summary Gantt chart. You identify the WEBVAR variable to the GANTT procedure using the HTML= option in the CHART statement and invoke the procedure using a WHERE clause to produce a Gantt chart of the top-level activities.

```
/* Create the Summary Gantt Chart with Drill Down Action */
pattern1  c=green  v=s;         /* Non-critical duration */
pattern2  c=green  v=e;         /* Slack duration        */
pattern3  c=red    v=s;         /* Critical duration     */
goptions cback=white htext=1.1;

title1 h=2 'Gantt Example 26';
```

```
title2 h=1.5 'Project Summary Gantt Chart';

proc gantt data=lanweb;
   id act wbs_code;
   where proj_lev=1;
   label act='SUBPROJECT' wbs_code='WBS CODE';
   chart / pcompress nojobnum
           duration=days
           mininterval=week scale=2.5
           mindate='30oct03'd maxdate='29feb04'd
           ref='30oct03:00:00'dt to '01mar04:00:00'dt by dtmonth
           reflabel
           html=webvar
           act=act succ=succ cprec=black;
   run;
```

The graph that is displayed when you click on one of the subprojects is determined by the name of the anchor that has been defined for the subproject. Before creating these graphs, you need to define the anchor name in an ODS HTML statement using the ANCHOR= option to add the anchor to the HTML body file. Since you have to create a chart for each subproject, you can automate this process by using a SAS macro.

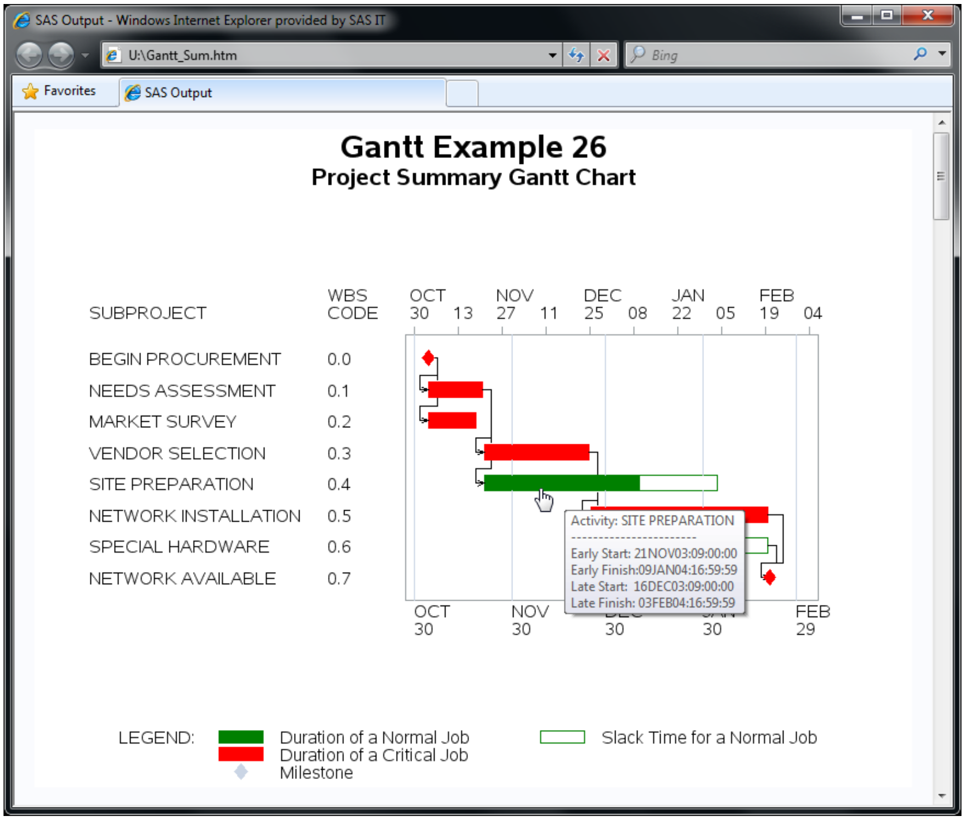
```
/* Define the macro to generate the detail charts */
%macro gandet(wbs);

goptions device=gif;
ods html anchor=&wbs;

title1 h=2 'Gantt Example 26';
title2 h=1.5 "Detail Gantt Chart for WBS="&wbs;

proc gantt data=lanweb;
   id act wbs_code;
   where index(wbs_code,&wbs)=1;
   label act='SUBPROJECT' wbs_code='WBS CODE';
   chart / pcompress nojobnum
           duration=days
           mininterval=week scale=2.5
           mindate='30oct03'd maxdate='29feb04'd
           ref='30oct03:00:00'dt to '01mar04:00:00'dt by dtmonth
           reflabel html=webvar2
           act=act succ=succ cprec=black;
   run;
%mend;

/* Generate each of the detail Gantt Charts */
%gandet('0.1');
%gandet('0.2');
%gandet('0.3');
%gandet('0.4');
%gandet('0.5');
%gandet('0.6');
```

Finally, use the ODS HTML CLOSE statement to close the body file and stop generating HTML output.

```
ods html close;
```

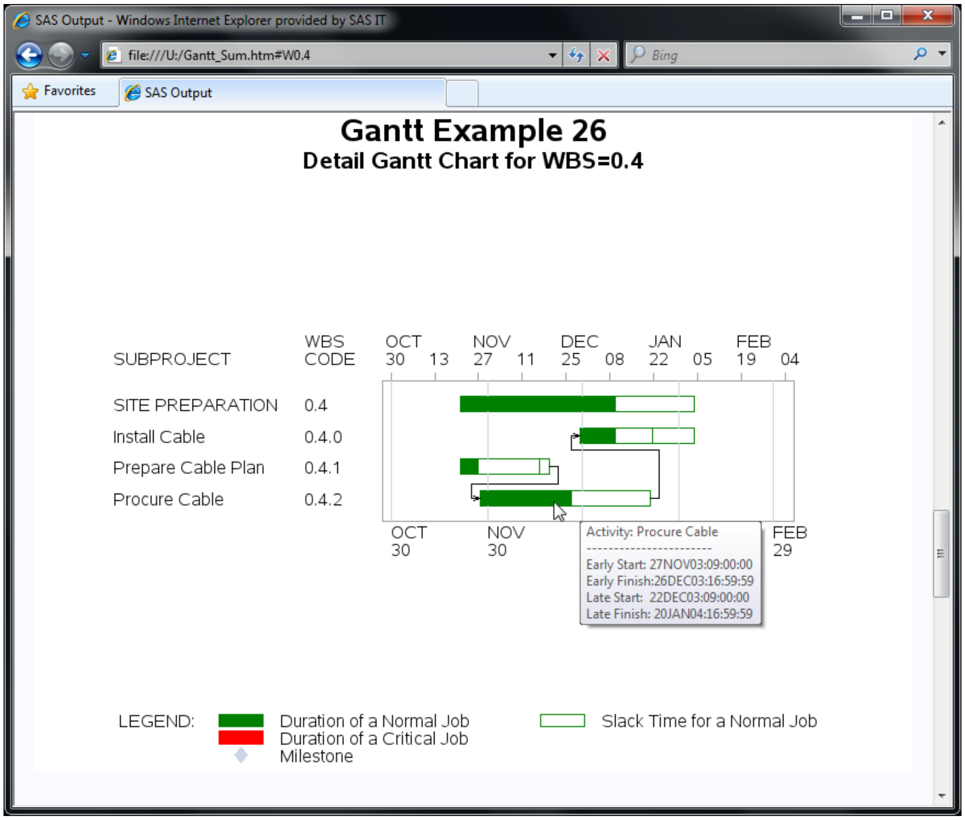
After you have closed the body file, you can display it in a browser window, as shown in Output 8.26.1, to view the output generated by this example.

Output 8.26.1 Summary Gantt Chart



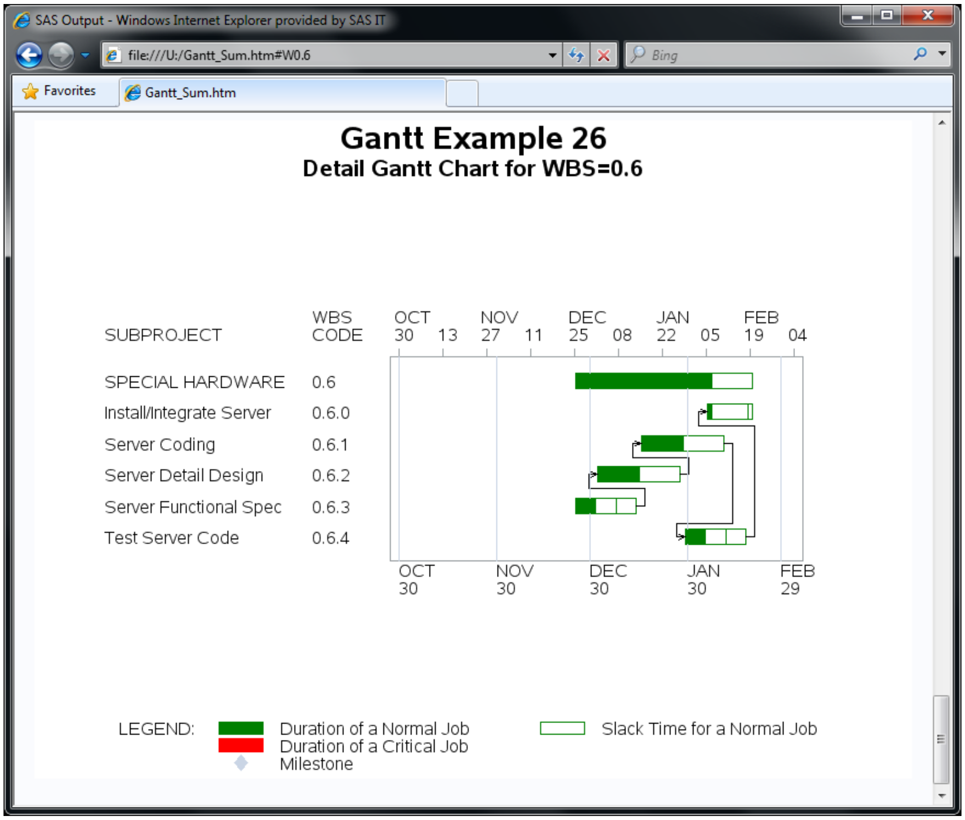
Notice the hand-shaped cursor on the SITE PREPARATION bar, which indicates that this bar is a "hot" link. The alternate text box displays the early and late schedules of the SITE PREPARATION activity. The status bar of the browser also shows that clicking the SITE PREPARATION bar will take you to the location identified by "Gantt_Sum.html#W0.4," which is shown in Output 8.26.2.

Output 8.26.2 Detail Gantt Chart for SITE PREPARATION



Similarly, the detail Gantt chart that is displayed when you click on the SPECIAL HARDWARE summary bar is shown in Output 8.26.3.

Output 8.26.3 Detail Gantt Chart for SPECIAL HARDWARE



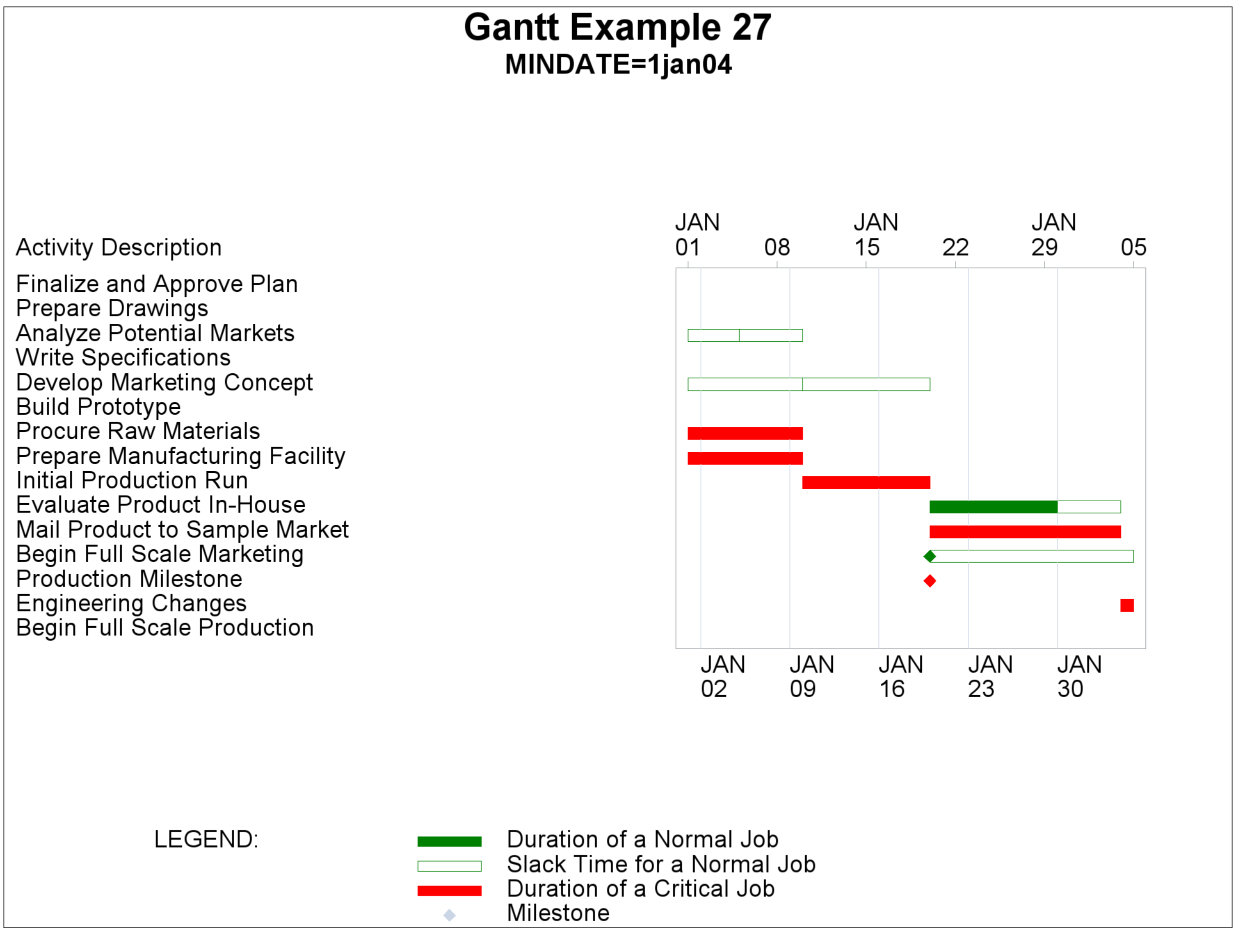
Example 8.27: Using the CHARTWIDTH= Option

This example illustrates the use of the CHARTWIDTH= option to create Gantt charts that are consistent in appearance. The data set used in this example is the SAVE data set created in Example 8.6.

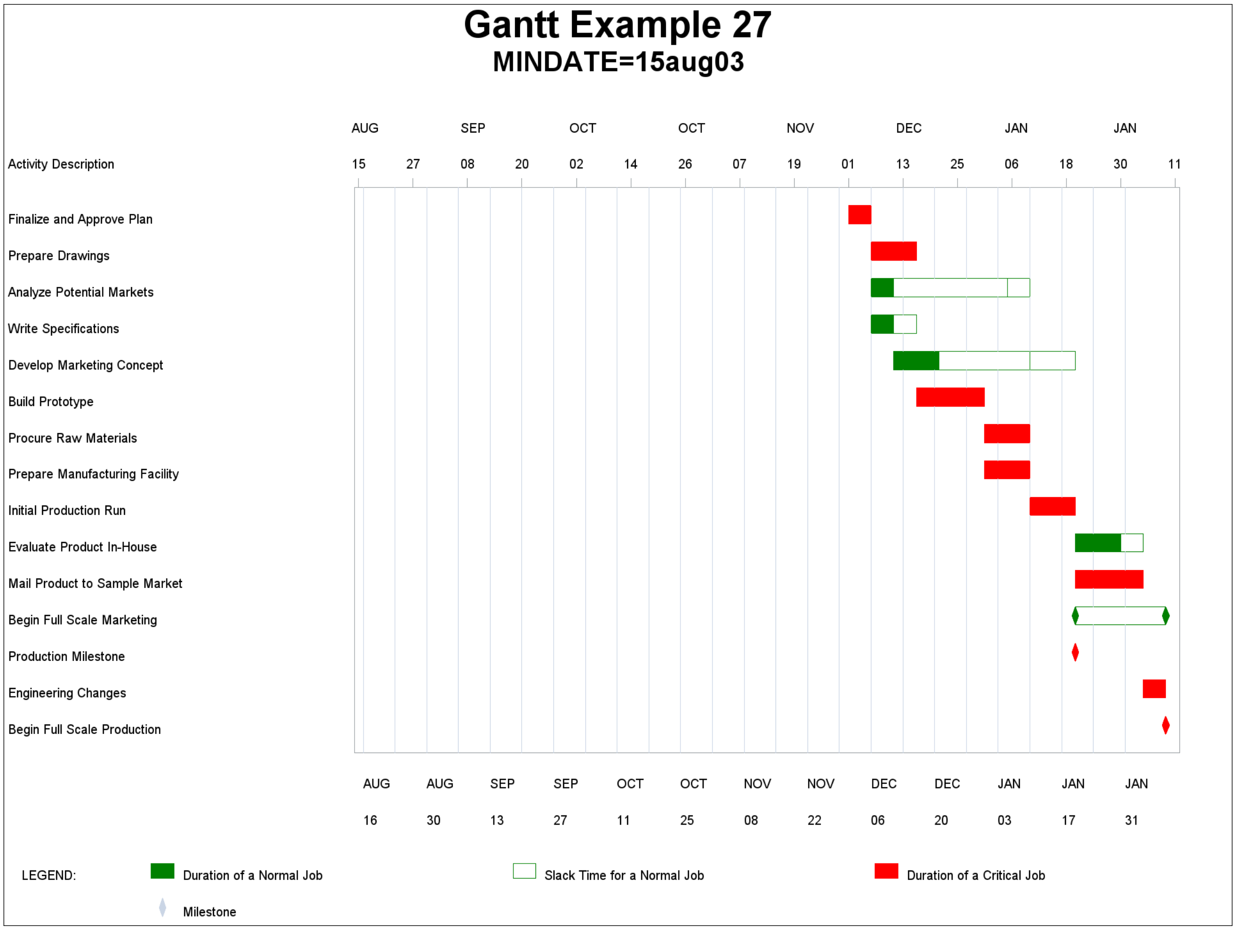
Gantt charts are first produced using different values of the MINDATE= option, and without specifying the CHARTWIDTH= option. Output 8.27.1 shows a Gantt chart using MINDATE='1jan04', and Output 8.27.2 shows a Gantt chart using MINDATE='15aug03'. Notice that the chart in Output 8.27.2 has a much larger chart area than the chart in Output 8.27.1, and the 'Activity Description' column is compressed and rather difficult to read.

```
proc gantt data=save;
   chart / mindate='1jan04'd maxdate='1feb04'd
           dur=days nojobnum compress height=2.0
           ref='2jan04'd to '2feb04'd by week
           reflabel;
   id descrpt;
   run;

proc gantt data=save;
   chart / mindate='15aug03'd maxdate='1feb04'd
           dur=days nojobnum compress height=2.0
           ref='16aug03'd to '2feb04'd by week
           reflabel;
   id descrpt;
   run;
```

Output 8.27.1 Without the CHARTWIDTH= Option (MINDATE=1Jan04)



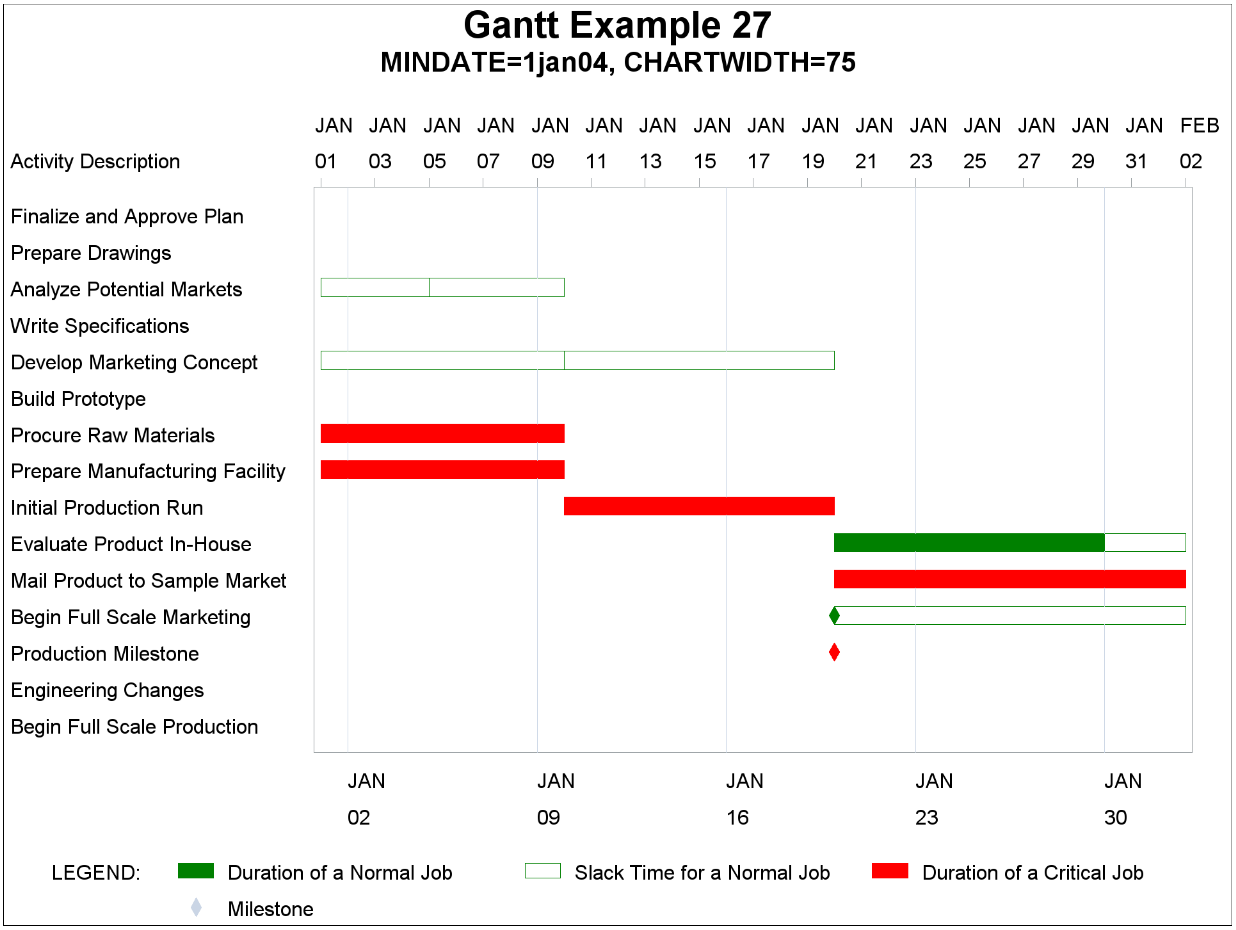
Output 8.27.2 Without the CHARTWIDTH= Option (MINDATE=15Aug03)



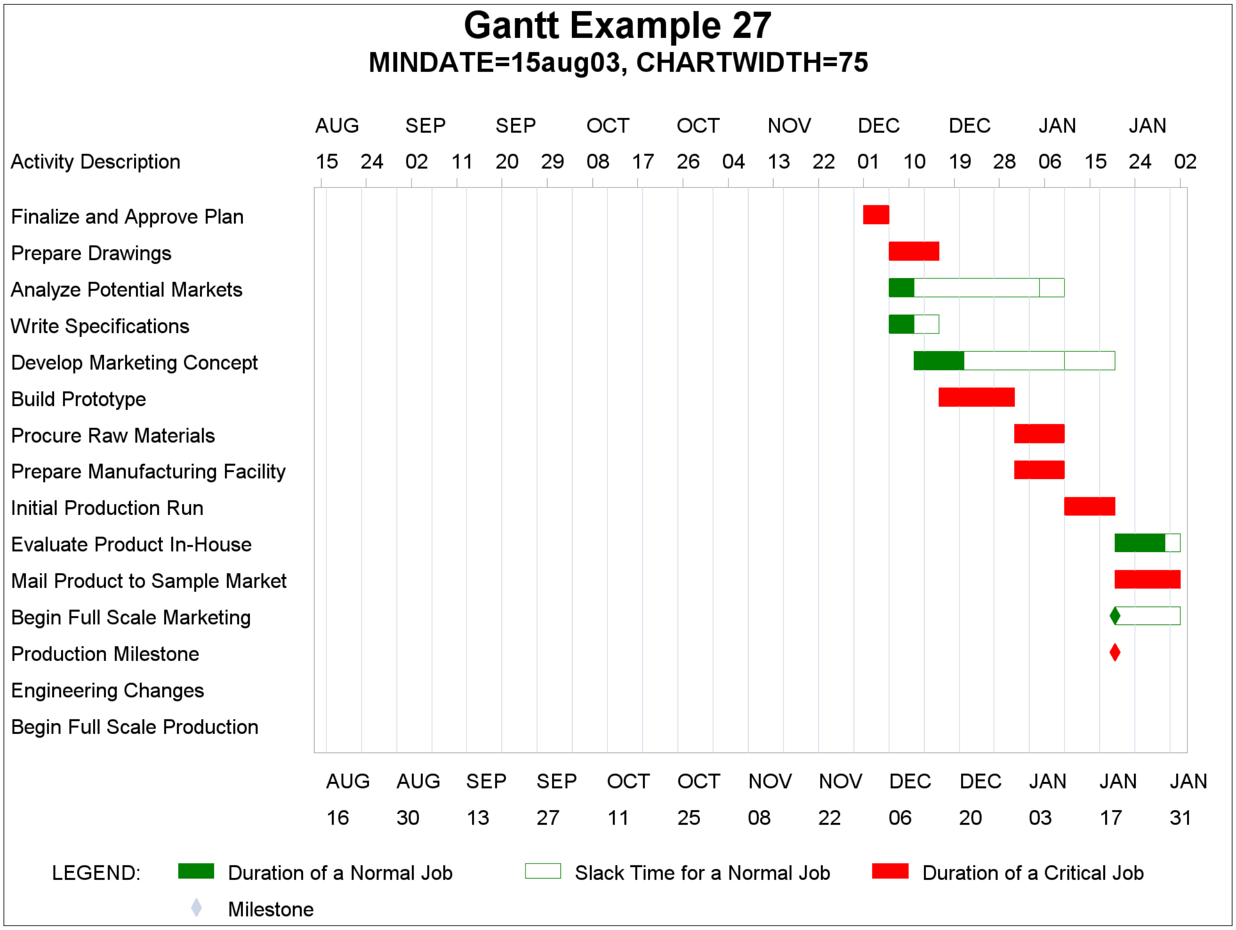
The same charts are now plotted with the CHARTWIDTH= option. The specification CHARTWIDTH=75 indicates that the chart is rescaled so the axis area is 75% of the chart width and the text area is 25% of the chart width. Therefore, specifying CHARTWIDTH=75 for both charts gives the two charts a consistent appearance. The output is shown in Output 8.27.3 and Output 8.27.4.

```
proc gantt data=save;
   chart / mindate='1jan04'd maxdate='1feb04'd
           dur=days nojobnum compress height=2.0
           ref='2jan04'd to '2feb04'd by week
           reflabel chartwidth=75;
   id descrpt;
   run;

proc gantt data=save;
   chart / mindate='15aug03'd maxdate='1feb04'd
           dur=days nojobnum compress height=2.0
           ref='16aug03'd to '2feb04'd by week
           reflabel chartwidth=75;
   id descrpt;
   run;
```

Output 8.27.3 Using the CHARTWIDTH= Option (MINDATE=1Jan04)



Output 8.27.4 Using the CHARTWIDTH= Option (MINDATE=15Aug03)



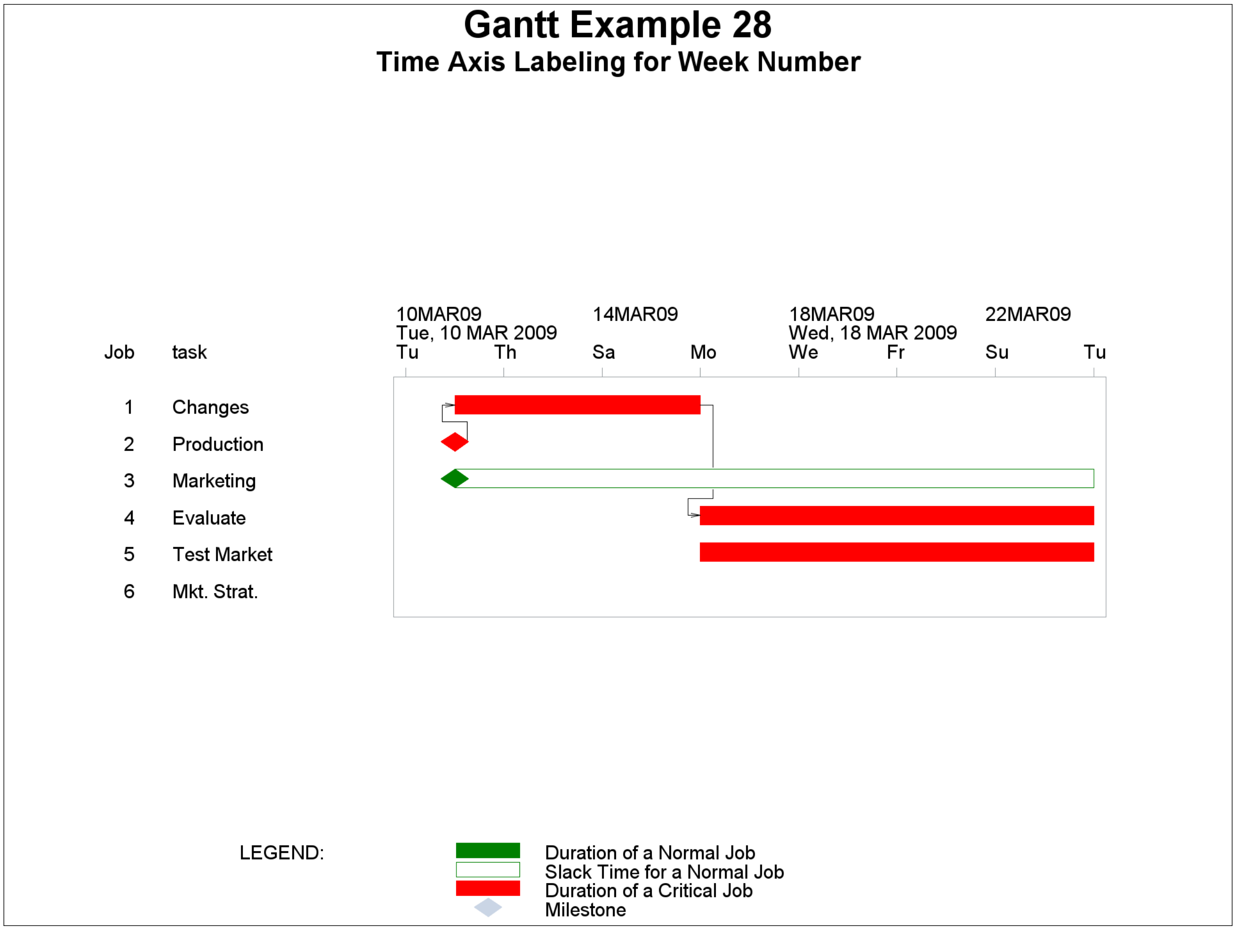
Example 8.28: Using the TIMEAXISFORMAT= Option

The following statements illustrate the use of the TIMEAXISFORMAT= option to specify formats for up to three rows of time-axis labels. The Activity data set for PROC CPM is the WIDGETA data set from Example 4.2, which defines the widget manufacturing project in AOA format.

```
* schedule the project subject to holidays and weekends;
proc cpm data=widgeta out=savehp
         date='11mar09'd;
   successor tail;
   activity head;
   duration days;
   id task dept descrpt;
run;

* sort the schedule by the early start date ;
proc sort;
   by e_start;
run;

* define a date format that includes the day of the week;
proc format;
    picture  dowdate (default=16) low-high = '%a, %d %b %Y'
        (datatype=date fill='0');
run;

* set up pattern statements;
pattern1 c=green v=s;    /* duration of a non-critical activity  */
pattern2 c=green v=e;    /* slack time for a noncrit. activity   */
pattern3 c=red v=s;      /* duration of a critical activity      */
pattern4 c=magenta v=e;  /* slack time for a supercrit. activity */
pattern5 c=magenta v=s;  /* duration of a supercrit. activity    */
pattern6 c=cyan v=s;     /* actual duration of an activity       */
pattern7 c=black v=e;    /* break due to a holiday               */
pattern8 c=blue v=s;     /* resource schedule of activity        */
pattern9 c=brown v=s;    /* baseline schedule of activity        */

* set graphics options;

* plot the logic Gantt chart using AOA representation;
proc gantt data=savehp (obs=6);
   chart / compress
           activity=head
           successor=tail
           mininterval=day
           increment=1
           dur=days
           maxdate='24MAR09'd
           taformat=(date7., dowdate., downame2.)
           cprec=black
           ;
   id task;
run;
```

The resulting Gantt chart, displayed in Output 8.28.1, contains one time-axis row for each format specified.

Output 8.28.1 Using the TIMEAXISFORMAT= Option



Statement and Option Cross-Reference Tables

The next two tables show which examples in this section use each of the statements and options in the GANTT procedure.

Table 8.15 Options Specified in Examples 4.1–4.14

Option	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A_FINISH=										X				
A_START=										X				
BETWEEN=		X												
CALID=									X					
CALENDAR=									X					
CAXIS=				X	X	X	X			X		X		
CHART var				X	X									
CHCON=										X				
CMILE=				X	X	X				X				
COMBINE												X		
COMPRESS					X		X			X		X		
CREF=							X							
CRITFLAG		X												
CTEXT=				X	X					X				
CTNOW=												X		
DATA=			X				X	X	X	X	X	X	X	X
DUR=				X	X	X	X	X		X	X	X	X	X
FILL		X												
FONT=								X		X				
HCONNECT										X				
HEIGHT=								X		X	X	X	X	X
HOLIDATA=			X					X	X	X	X	X	X	X
HOLIDAY=			X					X	X	X	X	X	X	X
HOLIDUR=								X						
HOLIFIN=			X						X	X	X	X		
ID		X	X	X	X	X	X	X	X	X	X	X	X	X
INTERVAL=								X						
LINEPRINTER	X	X												
L_FINISH=														X
LREF=							X							
L_START=														X
MARKBREAK								X	X					
MAXDATE=							X	X	X					
MINDATE=							X	X	X					
MININTERVAL=		X				X		X						
NOJOBNUM		X					X							
NOLEGEND		X				X								
PCOMPRESS								X	X		X		X	X
REF=		X				X	X							

Table 8.15 (continued)

Option	1	2	3	4	5	6	7	8	9	10	11	12	13	14
REFLABEL							X							
SCALE=		X				X			X		X			
S_FINISH=														X
SKIP=		X												
S_START=														X
SUMMARY		X												
TIMENOW=											X	X		
WORKDATA=									X					

Table 8.16 Options Specified in Examples 4.15–4.28

Option	15	16	17	18	19	20	21	22	23	24	25	26	27	28
A_FINISH=				X										
A_START=				X										
ACTIVITY=				X		X			X			X		X
ANNOTATE=							X							
BY	X	X												
CAXIS=			X	X	X	X	X		X					
CHARTWIDTH=													X	
CMILE=			X	X	X	X	X							
COMPRESS			X	X	X	X				X			X	X
CPREC=				X	X	X			X			X		X
CREF=						X	X							
CTEXT=									X					
CZONE=											X			
DATA=		X	X	X	X	X	X	X	X	X		X	X	X
DUR=	X		X	X	X	X	X	X			X	X	X	X
FONT=									X					
HEAD=					X									
HEIGHT=	X		X		X	X		X	X				X	
HOLIDATA=			X	X	X	X	X							
HOLIDAY=			X	X	X	X	X							
HOLIFIN=				X		X								
HTML=												X		
HTOFF=			X											
ID	X	X	X	X	X	X	X		X	X	X	X	X	X
INCREMENT=					X		X		X					X
INTERVAL=							X							
LABDATA=								X	X	X				
LABSPLIT=								X		X				
LABVAR=								X	X					
LAG=						X								
LEVEL=				X										
LREF=						X	X	X						

Table 8.16 (continued)

Option	15	16	17	18	19	20	21	22	23	24	25	26	27	28
MAXDATE=	X				X		X	X	X		X	X	X	X
MAXDEC=								X						
MAXDISLV=					X									
MINDATE=	X						X		X		X	X	X	
MININTERVAL=								X				X		X
MININTGV=					X	X								
MINOFFGV=					X				X					
MINOFFLV=					X	X			X					
NOJOBNUM								X	X	X		X	X	
NOLEGEND								X		X				
ONEZONEVAL											X			
PCOMPRESS	X	X					X	X	X		X	X		
PRECDATA=						X								
REF=						X	X	X		X		X	X	
REFLABEL						X						X	X	
SCALE=	X							X	X	X	X	X		
S_FINISH=										X				
SKIP=								X		X				
SPLIT=	X										X			
S_START=										X				
SUCCESSOR=				X		X			X			X		X
TAIL=					X									
TAFORMAT=														X
WPREC=						X			X					
ZONE=											X			

References

Bostwick, S. (1986). “Software Helps Solve LAN Selection Puzzle.” *Mini-Micro Systems* (February 14): 20–22.

Kuhfeld, W. F. (2010). *Statistical Graphics in SAS: An Introduction to the Graph Template Language and the Statistical Graphics Procedures*. Cary, NC: SAS Institute Inc.

Moder, J. J., Phillips, C. R., and Davis, E. W. (1983). *Project Management with CPM, PERT, and Precedence Diagramming*. New York: Van Nostrand Reinhold.

Chapter 9
The NETDRAW Procedure

Contents

Overview: NETDRAW Procedure

The NETDRAW procedure draws a network diagram of the activities in a project. Boxes (or nodes) are used to represent the activities, and lines (or arcs) are used to show the precedence relationships among the activities. Though the description of the procedure is written using project management terminology, PROC NETDRAW can be used to draw any network such as an organizational chart or a software flow diagram. The only information required by the procedure for drawing such a diagram is the name of each activity in the project (or node in the network) and a list of all its immediate successor activities (or nodes connected to it by arcs). Note that project networks are acyclic. However, the procedure can also be used to draw cyclic networks by specifying explicitly the coordinates for the nodes or by requesting the procedure to break the cycles in an arbitrary fashion.

The ACTNET statement in the NETDRAW procedure is designed to draw activity networks that represent a project in Activity-On-Node (AON) format. All network information is contained in SAS data sets. The input data sets used by PROC NETDRAW and the output data set produced by the procedure are as follows:

- The Network input data set contains the precedence information, namely, the activity-successor information for all the nodes in the network. This data set can be an Activity data set that is used as input to the CPM procedure or a Schedule data set that is produced by the CPM procedure, or it can even be a Layout data set produced by the NETDRAW procedure. The minimum amount of information that is required by PROC NETDRAW is the activity-successor information that can be obtained from any one of the preceding three possible types of data sets. The additional information in the input data set can be used by the procedure to add detail to the nodes in the diagram, and, in the case of the Layout data set, the procedure can use the _X_ and _Y_ variables to lay out the nodes and arcs of the diagram.

- The Annotate input data set contains the graphics and text that are to be annotated on the network diagram. This data set is used by the procedure through the Annotate facility in SAS/GRAPH software.

- The Layout output data set produced by PROC NETDRAW contains all the information about the layout of the network. For each node in the network, the procedure saves the (x, y) coordinates; for each arc between each pair of nodes, the procedure saves the (x, y) coordinates of each turning point of the arc in a separate observation. Using these values, the procedure can draw the network diagram without recomputing node placement and arc routing.

Two issues arise in drawing and displaying a network diagram: the layout of the diagram and the format of the display. The layout of the diagram consists of placing the nodes of the network and routing the arcs of the network in an appropriate manner. The format of the display includes the size of the nodes, the distance between nodes, the color of the nodes and arcs, and the information that is placed within each node. Several options available in the ACTNET statement enable you to control the format of the display and the layout of the diagram; these options and their uses are explained in detail later in this chapter.

Following is a list of some of the key aspects of the procedure:

- The Network input data set specifies the activities (or nodes) in the network and their immediate successors. The amount of information displayed within each node can be controlled by the ID= option and by the use of default variables in the data set.
- The procedure uses the node-successor information to determine the placement of the nodes and the layout of the arcs connecting the nodes.
- By default, PROC NETDRAW uses the topological ordering of the activity network to determine the x coordinates of the nodes. In a time-based network diagram, the nodes can be ordered according to any numeric, SAS date, time, or datetime variable (the ALIGN= variable) in the input data set.
- The network does not have to represent a project. You can use PROC NETDRAW to draw any network. If the network has no cycles, then the procedure bases the node placement and arc routing on the precedence relationships. Alternately, you can specify explicitly the node positions or use the ALIGN= variable, and let the procedure determine the arc routing.
- To draw networks with cycles, use the BREAKCYCLE option. Alternately, you can use the ALIGN= option or specify the node positions so that the procedure needs only to determine the arc routing. See Example 9.12 for an illustration of a cyclic network.
- The ZONE= option enables you to divide the network into horizontal bands or zones. This is useful in grouping the activities of the project according to some appropriate classification.
- The TREE option instructs PROC NETDRAW to check if the network is indeed a tree, and, if so, to exploit the tree structure in the node layout. This feature is useful for drawing organizational charts, hierarchical charts, and work break-down structures.
- PROC NETDRAW gives you the option of displaying the network diagram in one of three modes: graphics, line-printer, or full-screen. The default mode is graphics mode, which enables you to produce charts of high resolution quality. Graphics mode requires SAS/GRAPH software. See the section "Graphics Options" on page 681 for more information about producing high-resolution quality network diagrams. You can also produce line-printer quality network diagrams by specifying the LINEPRINTER (LP) option in the PROC NETDRAW statement. In addition to sending the output to either a plotter or printer, you can view the network diagram at the terminal in full-screen mode by specifying the FULLSCREEN (FS) option in the PROC NETDRAW statement. See the section "Full-Screen Options" on page 680 for more information about viewing network diagrams in full-screen mode.
- The full-screen version of the procedure enables you to move the nodes around on the screen (subject to maintaining the precedence order of the activities) and thus change the layout of the network diagram.
- The graphics version of the procedure enables you to annotate the network diagram using the Annotate facility in SAS/GRAPH software.

- The positions of the nodes and arcs of the layout determined by PROC NETDRAW are saved in an output data set called the Layout data set. This data set can be used again as input to PROC NETDRAW; using such a data set saves some processing time because the procedure does not need to determine the node and arc placement.
- If necessary, the procedure draws the network across page boundaries. The number of pages that are used depends on the number of print positions that are available in the horizontal and vertical directions.
- In graphics mode, the COMPRESS and PCOMPRESS options enable you to produce the network on one page. You can also control the number of pages used to create the network diagram with the HPAGES= and VPAGES= options.
- In graphics mode, the ROTATE and ROTATETEXT options enable you to produce a top-down tree diagram.

Getting Started: NETDRAW Procedure

The first step in defining a project is to make a list of the activities in the project and determine the precedence constraints that need to be satisfied by these activities. It is useful at this stage to view a graphical representation of the project network. In order to draw the network, you specify the nodes of the network and the precedence relationships among them. Consider the software development project that is described in the "Getting Started" section of Chapter 4, "The CPM Procedure." The network data are in the SAS data set SOFTWARE, displayed in Figure 9.1.

Figure 9.1 Software Project

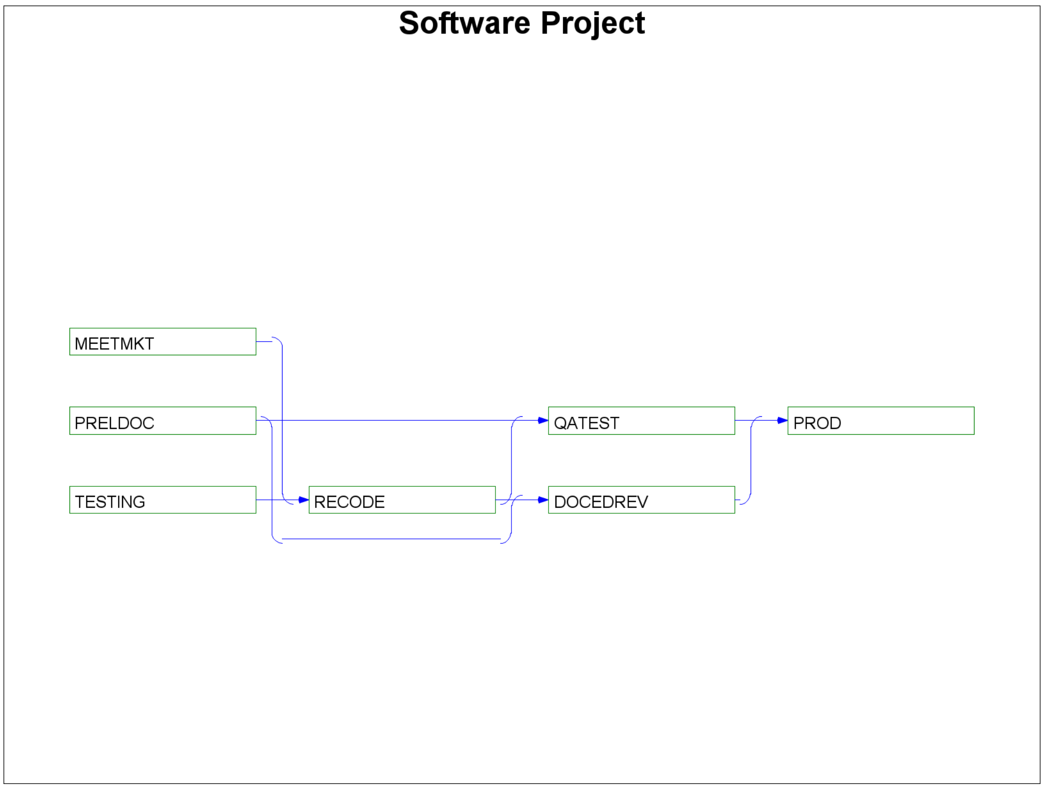
Software Project
Data Set SOFTWARE

Obs	descrpt	duration	activity	succesr1	succesr2
1	Initial Testing	20	TESTING	RECODE	
2	Prel. Documentation	15	PRELDOC	DOCEDREV	QATEST
3	Meet Marketing	1	MEETMKT	RECODE	
4	Recoding	5	RECODE	DOCEDREV	QATEST
5	QA Test Approve	10	QATEST	PROD	
6	Doc. Edit and Revise	10	DOCEDREV	PROD	
7	Production	1	PROD		

The following code produces the network diagram shown in Figure 9.2:

```
pattern1 v=e c=green;
pattern2 v=e c=red;

title h=3 'Software Project';
proc netdraw graphics data=software;
   actnet / act=activity htext=2
            succ=(succesr1 succesr2)
            pcompress separatearcs;
   run;
```

Figure 9.2 Software Project



The procedure determines the placement of the nodes and the routing of the arcs on the basis of the topological ordering of the nodes and attempts to produce a compact diagram. You can control the placement of the nodes by specifying explicitly the node positions. The data set SOFTNET, shown in Figure 9.3, includes the variables _X_ and _Y_, which specify the desired node coordinates. Note that the precedence information is conveyed using a single SUCCESSOR variable unlike the data set SOFTWARE, which contains two SUCCESSOR variables.

Figure 9.3 Software Project: Specify Node Positions

Software Project
Data Set SOFTNET

Obs	descrpt	duration	activity	succesor	_x_	_y_
1	Initial Testing	20	TESTING	RECODE	1	1
2	Meet Marketing	1	MEETMKT	RECODE	1	2
3	Prel. Documentation	15	PRELDOC	DOCEDREV	1	3
4	Prel. Documentation	15	PRELDOC	QATEST	1	3
5	Recoding	5	RECODE	DOCEDREV	2	2
6	Recoding	5	RECODE	QATEST	2	2
7	QA Test Approve	10	QATEST	PROD	3	3
8	Doc. Edit and Revise	10	DOCEDREV	PROD	3	1
9	Production	1	PROD		4	2

The following code produces a network diagram (shown in Figure 9.4) with the new node placement:

```
title h=3 'Software Project';
title2 h=2 'Controlled Layout';
proc netdraw graphics data=softnet;
   actnet / act=activity htext=1.25
            succ=(succesor)
            pcompress;
   run;
```

Figure 9.4 Software Project: Controlled Layout

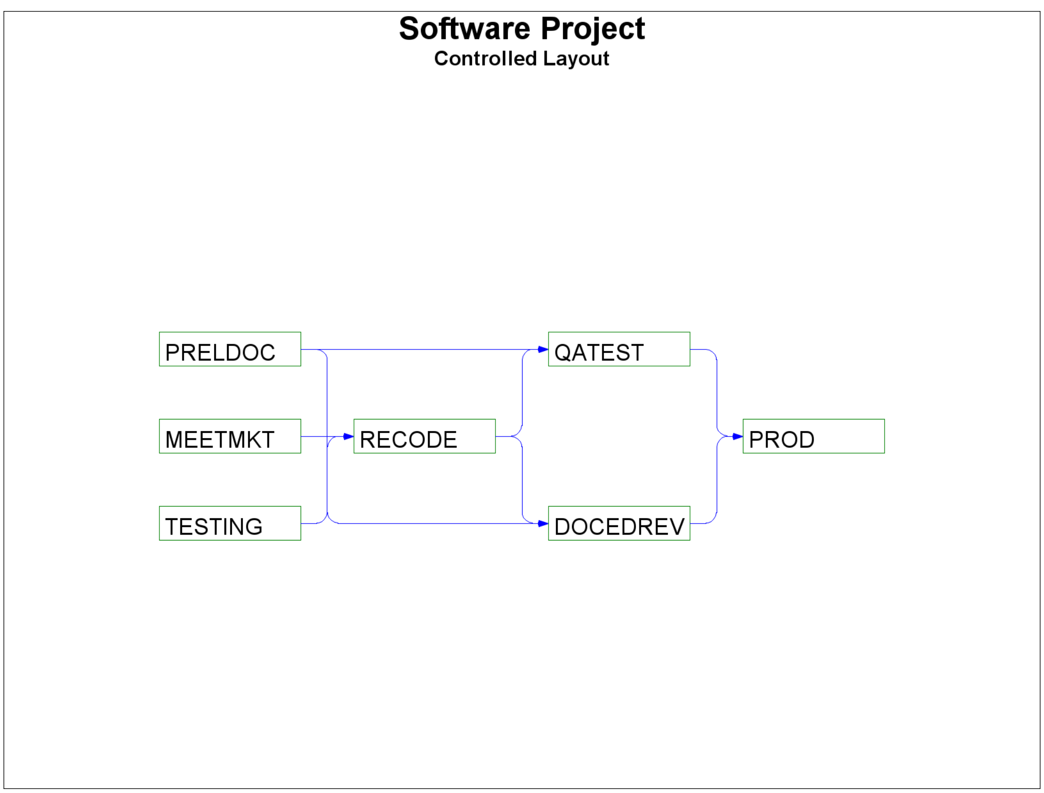


Figure 9.5 Software Project Schedule

Software Project
Project Schedule

descrpt	activity	succesr1	succesr2	duration	E_START
Initial Testing	TESTING	RECODE		20	01MAR04
Prel. Documentation	PRELDOC	DOCEDREV	QATEST	15	01MAR04
Meet Marketing	MEETMKT	RECODE		1	01MAR04
Recoding	RECODE	DOCEDREV	QATEST	5	21MAR04
QA Test Approve	QATEST	PROD		10	26MAR04
Doc. Edit and Revise	DOCEDREV	PROD		10	26MAR04
Production	PROD			1	05APR04

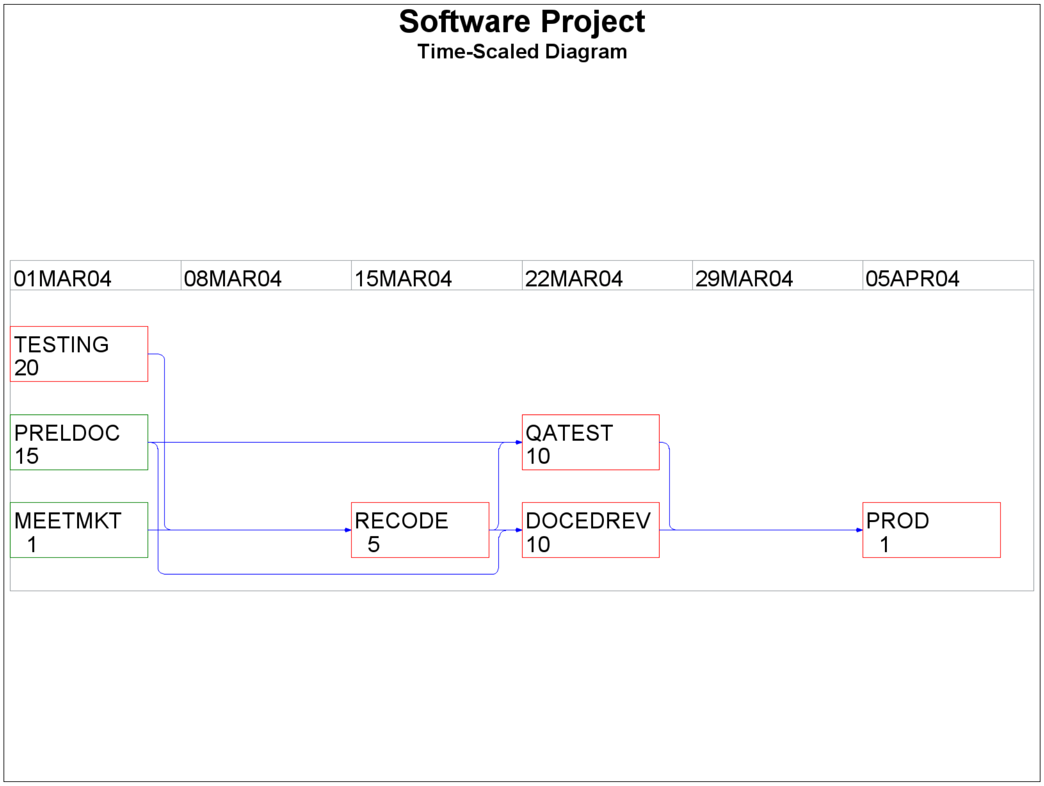
descrpt	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Initial Testing	20MAR04	01MAR04	20MAR04	0	0
Prel. Documentation	15MAR04	11MAR04	25MAR04	10	10
Meet Marketing	01MAR04	20MAR04	20MAR04	19	19
Recoding	25MAR04	21MAR04	25MAR04	0	0
QA Test Approve	04APR04	26MAR04	04APR04	0	0
Doc. Edit and Revise	04APR04	26MAR04	04APR04	0	0
Production	05APR04	05APR04	05APR04	0	0

While the project is in progress, you may want to use the network diagram to show the current status of each activity as well as any other relevant information about each activity. PROC NETDRAW can also be used to produce a time-scaled network diagram using the schedule produced by PROC CPM. The schedule data for the software project described earlier are saved in a data set, INTRO1, which is shown in Figure 9.5.

To produce a time-scaled network diagram, use the TIMESCALE option in the ACTNET statement, as shown in the following program. The MININTERVAL= and the LINEAR options are used to control the time axis on the diagram. The ID=, NOLABEL, and NODEFID options control the amount of information displayed within each node. The resulting diagram is shown in Figure 9.6.

```
title h=3 'Software Project';
title2 h=2 'Time-Scaled Diagram';
proc netdraw graphics data=intro1;
   actnet / act=activity succ=(succ:)
            separatearcs pcompress htext=2
            timescale linear frame mininterval=week
            id=(activity duration) nolabel nodefid;
   run;
```

Figure 9.6 Software Project: Time-Scaled Network Diagram



Several other options are available to control the layout of the nodes, the appearance of the network, and the format of the time axis. For projects that have natural divisions, you can use the ZONE= option to divide the network into horizontal zones or bands. For networks that have an embedded tree structure, you can use the TREE option to draw the network like a tree laid out from left to right, with the root at the left edge of the diagram; in graphics mode, you can obtain a top-down tree with the root at the top of the diagram. For cyclic networks you can use the BREAKCYCLE option to enable the procedure to break cycles. All of these options are discussed in detail in the following sections.

Syntax: NETDRAW Procedure

The following statements are used in PROC NETDRAW:

PROC NETDRAW *options* **;**
  ACTNET / *options* **;**

Functional Summary

Table 9.1 outlines the options available for the NETDRAW procedure classified by function.

Table 9.1 Functional Summary

Description	Statement	Option
Color Options		
Specifies the color of arcs	ACTNET	CARCS=
Specifies the color of time axis	ACTNET	CAXIS=
Specifies the fill color for critical nodes	ACTNET	CCNODEFILL=
Specifies the color of critical arcs	ACTNET	CCRITARCS=
Specifies the outline color of critical nodes	ACTNET	CCRITOUT=
Specifies the fill color for nodes	ACTNET	CNODEFILL=
Specifies the outline color of nodes	ACTNET	COUTLINE=
Specifies the color of reference lines	ACTNET	CREF=
Specifies the color of reference break lines	ACTNET	CREFBRK=
Specifies the text color	ACTNET	CTEXT=
Data Set Specifications		
Specifies the Annotate data set	ACTNET	ANNOTATE=
Specifies the Annotate data set	NETDRAW	ANNOTATE=
Specifies the Activity data set	NETDRAW	DATA=
Specifies the Network output data set	NETDRAW	OUT=
Format Control Options		
Specifies the height of node in character cells	ACTNET	BOXHT=
Specifies the width of node in character cells	ACTNET	BOXWIDTH=
Specifies the DURATION variable	ACTNET	DURATION=
Specifies the ID variables	ACTNET	ID=
Suppresses default ID variables	ACTNET	NODEFID
Suppresses ID variable labels	ACTNET	NOLABEL
Specifies the upper limit on number of pages	ACTNET	PAGES=
Indicates completed or in-progress activities	ACTNET	SHOWSTATUS
Specifies the horizontal distance between nodes	ACTNET	XBETWEEN=
Specifies the vertical distance between nodes	ACTNET	YBETWEEN=
Full-Screen Options		
Specifies the reference break line character	ACTNET	BRKCHAR=
Specifies the characters for node outlines and connections	ACTNET	FORMCHAR=
Specifies the reference character	ACTNET	REFCHAR=
Graphics Catalog Options		
Specifies the description for the catalog entry	ACTNET	DESCRIPTION=
Specifies the name for the catalog entry	ACTNET	NAME=
Specifies the name of the graphics catalog	NETDRAW	GOUT=
Graphics Display Options		
Specifies the length of arrowhead in character cells	ACTNET	ARROWHEAD=
Centers each ID variable within node	ACTNET	CENTERID
Compresses the diagram to a single page	ACTNET	COMPRESS

Table 9.1 *continued*

Description	Statement	Option
Specifies the text font	ACTNET	FONT=
Specifies the text height	ACTNET	HEIGHT=
Specifies the horizontal margin in character cells	ACTNET	HMARGIN=
Specifies the number of horizontal pages	ACTNET	HPAGES=
Specifies the reference line style	ACTNET	LREF=
Specifies the reference break line style	ACTNET	LREFBRK=
Specifies the width of lines used for critical arcs	ACTNET	LWCRIT=
Specifies the width of lines	ACTNET	LWIDTH=
Specifies the width of outline for nodes	ACTNET	LWOUTLINE=
Suppresses filling of arrowheads	ACTNET	NOARROWFILL
Suppresses page number	ACTNET	NOPAGENUMBER
Suppresses vertical centering	ACTNET	NOVCENTER
Specifies the number of nodes in horizontal direction	ACTNET	NXNODES=
Specifies the number of nodes in vertical direction	ACTNET	NYNODES=
Displays page number at upper right corner	ACTNET	PAGENUMBER
Specifies the PATTERN variable	ACTNET	PATTERN=
Proportionally compresses the diagram	ACTNET	PCOMPRESS
Draws arcs with rectangular corners	ACTNET	RECTILINEAR
Reverses the order of the y pages	ACTNET	REVERSEY
Rotates the network diagram	ACTNET	ROTATE
Rotates text within node by 90 degrees	ACTNET	ROTATETEXT
Draws arcs along distinct tracks	ACTNET	SEPARATEARCS
Specifies the vertical margin in character cells	ACTNET	VMARGIN=
Specifies the number of vertical pages	ACTNET	VPAGES=
Layout Options		
Breaks cycles in cyclic networks	ACTNET	BREAKCYCLE
Uses dynamic programming algorithm to route arcs	ACTNET	DP
Specifies the number of horizontal tracks between nodes	ACTNET	HTRACKS=
Routes arcs along potential node positions	ACTNET	NODETRACK
Disables use of dynamic programming to route arcs	ACTNET	NONDP
Blocks track along potential node positions	ACTNET	NONODETRACK
Restricts scope of arc layout algorithm	ACTNET	RESTRICTSEARCH
Uses spanning tree layout	ACTNET	SPANNINGTREE
Draws network as a tree, if possible	ACTNET	TREE
Specifies the number of vertical tracks between nodes	ACTNET	VTRACKS=
Line-Printer Options		
Specifies the reference break line character	ACTNET	BRKCHAR=
Specifies the characters for node outlines and connections	ACTNET	FORMCHAR=
Specifies the reference character	ACTNET	REFCHAR=
Mode Options		
Invokes full-screen version	NETDRAW	FULLSCREEN

Table 9.1 *continued*

Description	Statement	Option
Invokes graphics version	NETDRAW	GRAPHICS
Invokes line-printer version	NETDRAW	LINEPRINTER
Suppresses display of diagram	NETDRAW	NODISPLAY
Network Specifications		
Specifies the ACTIVITY variable	ACTNET	ACTIVITY=
Specifies the LAG variables	ACTNET	LAG=
Specifies the SUCCESSOR variables	ACTNET	SUCCESSOR=
Time-Scale Options		
Specifies the ALIGN variable	ACTNET	ALIGN=
Draws reference lines at every level	ACTNET	AUTOREF
Draws a border around the network diagram and axes	ACTNET	FRAME
Draws all vertical levels	ACTNET	LINEAR
Specifies the maximum number of empty columns between tick marks	ACTNET	MAXNULLCOLUMN=
Specifies the smallest interval per level	ACTNET	MININTERVAL=
Specifies the number of levels per tick mark	ACTNET	NLEVELSPERCOLUMN=
Suppresses time axis on continuation pages	ACTNET	NOREPEATAXIS
Omits the time axis	ACTNET	NOTIMEAXIS
Stops procedure if align value is missing	ACTNET	QUITMISSINGALIGN
Draws zigzag reference line at breaks	ACTNET	REFBREAK
Shows all breaks in time axis	ACTNET	SHOWBREAK
Draws time-scaled diagram	ACTNET	TIMESCALE
Uses format of variable and not default	ACTNET	USEFORMAT
Tree Options		
Centers each node with respect to subtree	ACTNET	CENTERSUBTREE
Specifies the order of the children of each node	ACTNET	CHILDORDER=
Separates the sons of a node for symmetry	ACTNET	SEPARATESONS
Uses spanning tree layout	ACTNET	SPANNINGTREE
Draws network as a tree, if possible	ACTNET	TREE
Web Options		
Specifies the Image map output data set	NETDRAW	IMAGEMAP=
Specifies the HTML variable	ACTNET	WEB=
Zone Options		
Divides network into connected components	ACTNET	AUTOZONE
Suppresses zone labels	ACTNET	NOZONELABEL
Specifies the ZONE variable	ACTNET	ZONE=
Labels zones	ACTNET	ZONELABEL
Sets missing pattern values using zone	ACTNET	ZONEPAT
Leaves extra space between zones	ACTNET	ZONESPACE

PROC NETDRAW Statement

PROC NETDRAW *options* **;**

The following options can appear in the PROC NETDRAW statement.

ANNOTATE=*SAS-data-set*

specifies the input data set that contains the appropriate annotate variables for the purpose of adding text and graphics to the network diagram. The data set specified must be an Annotate data set. See the section "Using the Annotate Facility" on page 703 for further details about this option.

DATA=*SAS-data-set*

names the SAS data set to be used by PROC NETDRAW for producing a network diagram. If DATA= is omitted, the most recently created SAS data set is used. This data set, also referred to as the Network data set, contains the network information (ACTIVITY and SUCCESSOR variables) and any ID variables that are to be displayed within the nodes. For details about this data set, see the section "Network Input Data Set" on page 687.

FULLSCREEN

FS

indicates that the network be drawn in full-screen mode. This enables you to view the network diagram produced by NETDRAW in different scales; you can also move nodes around the diagram to modify the layout.

GOUT=*graphics-catalog*

specifies the name of the graphics catalog used to save the output produced by PROC NETDRAW for later replay. This option is valid only if the GRAPHICS option is specified.

GRAPHICS

indicates that the network diagram produced be of high-resolution quality. If you specify the GRAPHICS option, but you do not have SAS/GRAPH software licensed at your site, the procedure stops and issues an error message. GRAPHICS is the default mode.

IMAGEMAP=*SAS-data-set*

names the SAS data set that receives a description of the areas of a graph and a link for each area. This information is for the construction of HTML image maps. You use a SAS DATA step to process the output file and generate your own HTML files. The graph areas correspond to the link information that comes from the WEB variable in the Network data set. This gives you complete control over the appearance and structure of your HTML pages.

LINEPRINTER

LP

produces a network diagram of line-printer quality.

NODISPLAY

requests the procedure not to display any output. The procedure still produces the Layout data set containing the details about the network layout. This option is useful to determine node placement and arc routing for a network that can be used at a later time to display the diagram.

OUT=*SAS-data-set*
specifies a name for the output data set produced by PROC NETDRAW. This data set, also referred to as the Layout data set, contains the node and arc placement information determined by PROC NETDRAW to draw the network. This data set contains all the information that was specified in the Network data set to define the project; in addition, it contains variables that specify the coordinates for the nodes and arcs of the network diagram. For details about the Layout data set, see the section "Layout Data Set" on page 694.

If the OUT= option is omitted, the procedure creates a data set and names it according to the DATA*n* convention.

ACTNET Statement

ACTNET / *options* **;**

The ACTNET statement draws the network diagram. You can specify several options in this statement to control the appearance of the network. All these options are described in the current section under appropriate headings: first, all options that are valid for all modes of the procedure are listed, followed by the options classified according to the mode (full-screen, graphics, or line-printer) of invocation of the procedure.

General Options

ACTIVITY=*variable*
specifies the variable in the Network data set that names the nodes in the network. If the data set contains a variable called _FROM_, this specification is ignored; otherwise, this option is required.

ALIGN=*variable*
specifies the variable in the Network data set containing the time values to be used for positioning each activity. This options triggers the TIMESCALE option that adds a time axis at the top of the network and aligns the nodes of the network according to the values of the ALIGN= variable. The minimum and maximum values of this variable are used to determine the time axis. The format of this variable is used to determine the default value of the MININTERVAL= option, which, in turn, determines the format of the time axis.

AUTOREF
draws reference lines at every tick mark. This option is valid only for time-scaled network diagrams.

AUTOZONE
enables automatic zoning (or dividing) of the network into connected components. This option is equivalent to defining an automatic zone variable that associates a tree number for each node. The tree number refers to a number assigned (by the procedure) to each distinct tree of a spanning tree of the network.

BREAKCYCLE
breaks cycles by reversing the back arcs of the network. The back arcs are determined by constructing an underlying spanning tree of the network. Once cycles are broken, the nodes of the network are laid out using a topological ordering of the new network formed from the original network by ignoring the back arcs. The back arcs are drawn after determining the network layout. Note that only the back arcs go from right to left.

BOXHT=*boxht*

specifies the height of the box (in character cell positions) used for denoting a node. If this option is not specified, the height of the box equals the number of lines required for displaying all of the ID variable values for any of the nodes. See the ROTATETEXT option (under "Graphics Options") for an exception.

BOXWIDTH=*boxwdth*

specifies the width of the box (in character cell positions) used for denoting a node. If this option is not specified, the width of the box equals the maximum number of columns required for displaying all of the ID variable values for any of the nodes. See the ROTATETEXT option (under "Graphics Options") for an exception.

CENTERSUBTREE

positions each node at the center of the subtree that originates from that node instead of placing it at the midpoint of its children (which is the default behavior). Note that the nodes are placed at integral positions along an imaginary grid, so the positioning may not be exactly at the center. This option is valid only in conjunction with the TREE option.

CHILDORDER=*order*

orders the children of each node when the network is laid out using either the TREE or the SPANNINGTREE option. The valid values for this option are TOPDOWN and BOTTOMUP for default orientation, and LEFTRGHT and RGHTLEFT for rotated networks (drawn with the ROTATETEXT option). The default is TOPDOWN.

DP

causes PROC NETDRAW to use a dynamic programming (DP) algorithm to route the arcs. This DP algorithm is memory and CPU-intensive and is not necessary for most applications.

DURATION=*variable*

specifies a variable that contains the duration of each activity in the network. This value is used only for displaying the durations of each activity within the node.

FRAME

encloses the drawing area with a border. This option is valid only for time-scaled or zoned network diagrams.

HTRACKS=*integer*

controls the number of arcs that are drawn horizontally through the space between two adjacent nodes. This option enables you to control the arc-routing algorithm. The default value is based on the maximum number of successors of any node.

ID=*(variables)*

specifies the variables in the Network data set that are displayed within each node. In addition to the ID variables, the procedure displays the ACTIVITY variable, the DURATION variable (if the DURATION= option was specified), and any of the following variables in the Network data set: E_START, E_FINISH, L_START, L_FINISH, S_START, S_FINISH, A_START, A_FINISH, T_FLOAT, and F_FLOAT. See Chapter 4, "The CPM Procedure," for a description of these variables. If you specify the NODEFID option, only the variables listed in the ID= option are displayed.

LAG=*variable*

LAG=*(variables)*

specifies the variables in the Network data set that identify the lag types of the precedence relationships between an activity and its successors. Each SUCCESSOR variable is matched with the corresponding LAG variable; that is, for a given observation, the *i*th LAG variable defines the relationship between the activities specified by the ACTIVITY variable and the *i*th SUCCESSOR variable. The LAG variables must be character type, and their values are expected to be specified as one of FS, SS, SF, or FF, which denote 'Finish-to-Start', 'Start-to-Start', 'Start-to-Finish', and 'Finish-to-Finish', respectively. You can also use the *keyword_duration_calendar* specification used by the CPM procedure, although PROC NETDRAW uses only the *keyword* information and ignores the lag *duration* and the lag *calendar*. If no LAG variables exist or if an unrecognized value is specified for a LAG variable, PROC NETDRAW interprets the lag as a 'Finish-to-Start' type.

This option enables the procedure to identify the different types of nonstandard precedence constraints (Start-to-Start, Start-to-Finish, and Finish-to-Finish) on graphics quality network diagrams by drawing the arcs from and to the appropriate edges of the nodes.

LINEAR

plots one column for every *mininterval* between the minimum and maximum values of the ALIGN= variable. By default, only those columns that contain at least one activity are displayed. This option is valid only for time-scaled network diagrams.

MAXNULLCOLUMN=*maxncol*

MAXEMPTY=*maxncol*

MAXZCOL=*maxncol*

MAXNCOL=*maxncol*

specifies the maximum number of empty columns between two consecutive nonempty columns. The default value for this option is 0. Note that specifying the LINEAR option is equivalent to specifying the MAXNULLCOLUMN= option to be infinity. This option is valid only for time-scaled network diagrams.

MININTERVAL=*mininterval*

specifies the smallest interval to be used per column of the network diagram. Thus, if MININTERVAL=DAY, each column is used to represent a day, and all activities that start on the same day are placed in the same column. The valid values for *mininterval* are SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QTR, and YEAR. The default value of *mininterval* is determined by the format of the ALIGN= variable. The tick labels are formatted on the basis of *mininterval*; for example, if *mininterval* is DAY, the dates are marked using the DATE7. format, and if *mininterval* is HOUR, the labels are formatted as TIME5. and so on. This option is valid only for time-scaled network diagrams.

NLEVELSPERCOLUMN=*npercol*

NPERCOL=*npercol*

contracts the time axis by specifying that activities that differ in ALIGN= value by less than *npercol* units of MININTERVAL can be plotted in the same column. The default value of *npercol* is 1. This option is valid only for time-scaled network diagrams.

NODEFID

indicates that the procedure need not check for any of the default ID variables in the Network data set; if this option is in effect, only the variables specified in the ID= option are displayed within each node.

NODETRACK

specifies that the arcs can be routed along potential node positions if there is a clear horizontal track to the left of the successor (or _TO_) node. This is the default option. To prevent the use of potential node positions, use the NONODETRACK option.

NOLABEL

suppresses the labels. By default, the procedure uses the first three letters of the variable name to label all the variables that are displayed within each node of the network. The only exception is the variable that is identified by the ACTIVITY= option.

NONDP

uses a simple heuristic to connect the nodes. The default mode of routing is NONDP, unless the HTRACKS= or VTRACKS= option (or both) are specified and set to a number that is less than the maximum number of successors. The NONDP option is faster than the DP option.

NONODETRACK

blocks the horizontal track along potential node positions. This option may lead to more turns in some of the arcs. The default is NODETRACK.

NOREPEATAXIS

displays the time axis only on the top of the chart and not on every page. This option is useful if the different pages are to be glued together to form a complete diagram. This option is valid only for time-scaled network diagrams.

NOTIMEAXIS

suppresses the display of the time axis and its labels. Note that the nodes are still placed according to the time scale, but no axis is drawn. This option is valid only for time-scaled network diagrams.

NOZONELABEL

NOZONEDESCR

omits the zone labeling and the dividing lines. The network is still divided into zones based on the ZONE variable, but there is no demarcation or labeling corresponding to the zones.

PAGES=*npages*

specifies the maximum number of pages to be used for the network diagram in graphics and line-printer modes. The default value is 100.

QUITMISSINGALIGN

stops processing if the ALIGN= variable has any missing values. By default, the procedure tries to fill in missing values using the topological order of the network. This option is valid only for time-scaled network diagrams.

REFBREAK

shows breaks in the time axis by drawing a zigzag line down the diagram just before the tick mark at the break. This option is valid only for time-scaled network diagrams.

RESTRICTSEARCH

RSEARCH

restricts the scope of the arc layout algorithm by restricting the area of search for the arc layout when the DP option is in effect; this is useful in reducing the computational complexity of the dynamic programming algorithm. By default, using the DP algorithm to route the arcs, the *y* coordinates of the arcs can range through the entire height of the network. The RESTRICTSEARCH option limits the *y* coordinates to the minimum and the maximum of the *y* coordinates of the node and its immediate successors.

SEPARATESONS

separates the children (immediate successors) of a given node by adding an extra space in the center whenever it is needed to enable the node to be positioned at integral (x, y) coordinates. For example, if a node has two children, placing the parent node at the midpoint between the two children requires the *y* coordinate to be noninteger, which is not allowed in the Layout data set. By default, the procedure positions the node at the same *y* level as one of its children. The SEPARATESONS option separates the two children by adding a dummy child in between, thus enabling the parent node to be centered with respect to its children. This option is valid only in conjunction with the TREE option.

SHOWBREAK

shows breaks in the time axis by drawing a jagged break in the time axis line just before the tick mark corresponding to the break. This option is valid only for time-scaled network diagrams.

SHOWSTATUS

uses the variable STATUS (if it exists) in the Network data set to determine if an activity is in-progress or completed. Note that the STATUS variable exists in the Schedule data set produced by PROC CPM when used with an ACTUAL statement. If there is no STATUS variable or if the value is missing, the procedure uses the A_FINISH and A_START values to determine the status of the activity. If the network is drawn in line-printer or full-screen mode, activities in progress are outlined with the letter *P* and completed activities are outlined with the letter *F*; in high-resolution graphics mode, in-progress activities are marked with a diagonal line across the node from the bottom left to the top right corner, while completed activities are marked with two diagonal lines.

SPANNINGTREE

uses a spanning tree to place the nodes in the network. This method typically results in a wider layout than the default. However, for networks that have totally disjoint pieces, this option separates the network into connected components (or disjoint trees). This option is not valid for time-scaled or zoned network diagrams, because the node placement dictated by the spanning tree may not be consistent with the zone or the tickmark corresponding to the node.

SUCCESSOR=*(variables)*

specifies the variables in the Network data set that name all the immediate successors of the node specified by the ACTIVITY variable. This specification is ignored if the data set contains a variable named _TO_. At least one SUCCESSOR variable must be specified if the data set does not contain a variable called _TO_.

TIMESCALE

indicates that the network is to be drawn using a time axis for placing the nodes. This option can be used to align the network according to default variables. If the TIMESCALE option is specified without the ALIGN= option, the procedure looks for default variables in the following order: E_START, L_START, S_START, and A_START. The first of these variables that is found is used as the ALIGN= variable.

TREE

TREELAYOUT

requests the procedure to draw the network as a tree if the network is indeed a tree (that is, all the nodes have at most one immediate predecessor). The option is ignored if the network does not have a tree structure.

USEFORMAT

indicates that the explicit format of the ALIGN= variable is to be used instead of the default format based on the MININTERVAL= option. Thus, for example, if the ALIGN variable contains SAS date values, by default, the procedure uses the DATE7. format for the time axis labels irrespective of the format of the ALIGN= variable. The USEFORMAT option specifies that the variable's format should be used for the labels instead of the default format. This option is valid only for time-scaled network diagrams.

VTRACKS=*integer*

controls the number of arcs that are drawn vertically through the space between two adjacent nodes. A default value is based on the maximum number of successors of any node.

XBETWEEN=*integer*

HBETWEEN=*integer*

specifies the horizontal distance (in character cell positions) between two adjacent nodes. The value for this option must be at least 3; the default value is 5.

YBETWEEN=*integer*

VBETWEEN=*integer*

specifies the vertical distance (in character cell positions) between two adjacent nodes. The value for this option must be at least 3; the default value is 5.

ZONE=*variable*

names the variable in the Network data set used to separate the network diagram into zones.

ZONELABEL

ZONEDESCR

labels the different zones and draws dividing lines between two consecutive zones. This is the default behavior; to omit the labels and the dividing lines, use the NOZONELABEL option.

ZONESPACE

ZONELEVADD

draws the network with an extra row between two consecutive zones.

Full-Screen Options

BRKCHAR=*brkchar*

specifies the character used for drawing the zigzag break lines down the chart at break points of the time axis. The default value is >. This option is valid only for time-scaled network diagrams.

CARCS=*color*
specifies the color of the connecting lines (or arcs) between the nodes. The default value of this option is CYAN.

CAXIS=*color*
specifies the color of the time axis. The default value is WHITE. This option is valid only for time-scaled network diagrams.

CCRITARCS=*color*
specifies the color of arcs connecting critical activities. The procedure uses the values of the E_FINISH and L_FINISH variables (if they are present) in the Network data set to determine the critical activities. The default value is the value of the CARCS= option.

CREF=*color*
specifies the color of the reference lines. The default value is WHITE. This option is valid only for time-scaled network diagrams.

CREFBRK=*color*
specifies the color of the lines drawn to denote breaks in the time axis. The default value is WHITE. This option is valid only for time-scaled network diagrams.

FORMCHAR [*index list***]=***'string'*
specifies the characters used for node outlines and arcs. See the section "Line-Printer Options" on page 687 for a description of this option.

PATTERN=*variable*
specifies an integer-valued variable in the Network data set that identifies the color number for each node of the network. If the data set contains a variable called _PATTERN, this specification is ignored. All the colors available for the full-screen device are used in order corresponding to the number specified in the PATTERN variable; if the value of the PATTERN variable is more than the number of colors available for the device, the colors are repeated starting once again with the first color. If a PATTERN variable is not specified, the procedure uses the first color for noncritical activities, the second color for critical activities, and the third color for supercritical activities.

REFCHAR=*refchar*
specifies the reference character used for drawing reference lines. The default value is "|". This option is valid only for time-scaled network diagrams.

ZONEPAT
indicates that if a PATTERN variable is not specified or is missing and if a ZONE= variable is present, then the node colors are based on the value of the ZONE= variable.

Graphics Options

ANNOTATE=*SAS-data-set*
specifies the input data set that contains the appropriate annotate variables for the purpose of adding text and graphics to the network diagram. The data set specified must be an Annotate data set. See the section "Using the Annotate Facility" on page 703 for further details about this option.

ARROWHEAD=*integer*

specifies the length of the arrowhead in character cell positions. You can specify ARROWHEAD = 0 to suppress arrowheads altogether. The default value is 1.

CARCS=*color*

specifies the color to use for drawing the connecting lines between the nodes. The default color depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 705 for more information.

CAXIS=*color*

specifies the color of the time axis. This option is valid only for time-scaled network diagrams. The default color depends on the GSTYLE system option and the value of the CTEXT= option; see the section "ODS Style Templates" on page 705 for more information.

CCNODEFILL=*color*

specifies the fill color for all critical nodes of the network diagram. If you specify this option, the procedure uses a solid fill pattern (with the color specified in this option) for all critical nodes, ignoring any fill pattern specified in the PATTERN statements; the PATTERN statements are used only to obtain the color of the outline for these nodes unless you specify the CCRITOUT= option. The default value for this option is the value of the CNODEFILL= option, if it is specified; otherwise, the procedure uses the PATTERN statements to determine the fill pattern and color.

CCRITARCS=*color*

specifies the color of arcs connecting critical activities. The procedure uses the values of the E_FINISH and L_FINISH variables (if they are present) in the Network data set to determine the critical activities. The default value of this option is the value of the CARCS= option.

CCRITOUT=*color*

specifies the outline color for critical nodes. The default value for this option is the value of the COUTLINE= option, if it is specified; otherwise, it is the same as the pattern color for the node.

CENTERID

centers the ID values placed within each node. By default, character valued ID variables are left justified and numeric ID variables are right justified within each node. This option centers the ID values within each node.

CNODEFILL=*color*

specifies the fill color for all nodes of the network diagram. If you specify this option, the procedure uses a solid fill pattern with the specified color, ignoring any fill pattern specified in the PATTERN statements; the PATTERN statements are used only to obtain the color of the outline for the nodes, unless you specify the COUTLINE= option.

COMPRESS

draws the network on one physical page. By default, the procedure draws the network across multiple pages if necessary, using a default scale that allots one character cell position for each letter within the nodes. Sometimes, to get a broad picture of the network and all its connections, you may want to view the entire network on one screen. If the COMPRESS option is specified, PROC NETDRAW determines the horizontal and vertical transformations needed so that the network is compressed to fit on one screen.

COUTLINE=*color*

specifies an outline color for all nodes. By default, the procedure sets the outline color for each node to be the same as the fill pattern for the node. This option is useful when used in conjunction with a solid fill using a light color. Note that if an empty fill pattern is specified, then the COUTLINE= option will cause all nodes to appear the same.

CREF=*color*

specifies the color of the reference lines. This option is valid only for time-scaled network diagrams. The default color depends on the GSTYLE system option and the value of the CTEXT= option; see the section "ODS Style Templates" on page 705 for more information.

CREFBRK=*color*

specifies the color of the zigzag break lines. This option is valid only for time-scaled network diagrams. The default color depends on the GSTYLE system option and the value of the CTEXT= option; see the section "ODS Style Templates" on page 705 for more information.

CTEXT=*color*

CT=*color*

specifies the color of all text on the network diagram including variable names or labels, values of ID variables, and so on. The default color depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 705 for more information.

DESCRIPTION=*'string'*

DES=*'string'*

specifies a descriptive string, up to 40 characters in length, that appears in the description field of the master menu in PROC GREPLAY. If the DESCRIPTION= option is omitted, the description field contains a description assigned by PROC NETDRAW.

FILLPAGE

causes the diagram on each page to be magnified (if necessary) to fill up the page.

FONT=*font*

specifies the font of the text. The default font depends on the GOPTIONS statement and the GSTYLE system option; see the section "ODS Style Templates" on page 705 for more information.

HEIGHT=*h*

HTEXT=*h*

specifies that the height for all text in PROC NETDRAW (excluding the titles and footnotes) be *h* times the value of the global HTEXT= option, which is the default text height specified in the GOPTIONS statement of SAS/GRAPH. The value of *h* must be a positive real number; the default value is 1.0.

HMARGIN=*integer*

specifies the width of a horizontal margin (in number of character cell positions) for the network in graphics mode. The default width is 1.

HPAGES=*h*

NXPAGES=*h*

specifies that the network diagram is to be produced using *h* horizontal pages. However, it may not be possible to use *h* horizontal pages due to intrinsic constraints on the output.

For example, PROC NETDRAW requires that every horizontal page should contain at least one *x* level. Thus, the number of horizontal pages can never exceed the number of vertical levels in the network. The exact number of horizontal pages used by the network diagram is given in the _ORNETDR macro variable. See the section "Macro Variable _ORNETDR" on page 704 for further details.

The appearance of the diagram with respect to the HPAGES= option is also influenced by the presence of other related procedure options. The HPAGES= option performs the task of determining the number of vertical pages in the absence of the VPAGES= option. If the COMPRESS or PCOMPRESS option is specified in this scenario, the chart uses one vertical page (unless the HPAGES= and VPAGES= options are specified). If neither the COMPRESS nor PCOMPRESS option is specified, the number of vertical pages is computed in order to display as much of the chart as possible in a proportional manner.

LREF=*linestyle*

specifies the linestyle (1-46) of the reference lines. The default linestyle is 1, a solid line. See Figure 8.5 in Chapter 8, "The GANTT Procedure," for examples of the various line styles available. This option is valid only for time-scaled network diagrams.

LREFBRK=*linestyle*

specifies the linestyle (1-46) of the zigzag break lines. The default linestyle is 1, a solid line. See Figure 8.5 in Chapter 8, "The GANTT Procedure," for examples of the various line styles available. This option is valid only for time-scaled network diagrams.

LWCRIT=*integer*

specifies the line width for critical arcs and the node outlines for critical activities. If the LWCRIT= option is not specified, the procedure uses the value specified for the LWIDTH= option.

LWIDTH=*integer*

specifies the line width of the arcs and node outlines. The default line width is 1.

LWOUTLINE=*integer*

specifies the line width of the node outlines. The default line width for the node outline is equal to LWIDTH for noncritical nodes and LWCRIT for critical nodes.

NAME=*'string'*

specifies a string of up to eight characters that appears in the name field of the catalog entry for the graph. The default name is NETDRAW. If either the name specified or the default name duplicates an existing name in the catalog, then the procedure adds a number to the duplicate name to create a unique name, for example, NETDRAW2.

NOARROWFILL

draws arrowheads that are not filled. By default, the procedure uses filled arrowheads.

NOPAGENUMBER

NONUMBER

suppresses the page numbers that are displayed in the top right corner of each page of a multipage network diagram. Note that the pages are ordered from left to right, bottom to top (unless the REVERSEY option is specified).

NOVCENTER

draws the network diagram just below the titles without centering in the vertical direction.

NXNODES=*nx*

specifies the number of nodes that should be displayed horizontally across each page of the network diagram. This option determines the value of the HPAGES= option; this computed value of HPAGES overrides the specified value for the HPAGES= options.

NYNODES=*ny*

specifies the number of nodes that should be displayed vertically across each page of the network diagram. This option determines the value of the VPAGES= option; this computed value of VPAGES overrides the specified value for the VPAGES= options.

PAGENUMBER

PAGENUM

numbers the pages of the network diagram on the top right-hand corner of the page if the diagram exceeds one page. The numbering scheme is from left to right, bottom to top (unless the REVERSEY option is specified).

PATTERN=*variable*

specifies an integer-valued variable in the Network data set that identifies the pattern for filling each node of the network. If the data set contains a variable called _PATTERN, this specification is ignored. The patterns are assumed to have been specified using PATTERN statements. If a PATTERN variable is not specified, the procedure uses the first PATTERN statement for noncritical activities, the second PATTERN statement for critical activities, and the third PATTERN statement for supercritical activities.

PCOMPRESS

draws the network diagram on one physical page. As with the COMPRESS option, the procedure determines the horizontal and vertical transformation needed so that the network is compressed to fit on one screen. However, in this case, the transformations are such that the network diagram is proportionally compressed. See Example 9.4 for an illustration of this option.

If the HPAGES= and VPAGES= options are used to control the number of pages, each page of the network diagram is drawn while maintaining the original aspect ratio.

RECTILINEAR

draws arcs with rectangular corners. By default, the procedure uses rounded turning points and rounded arc merges in graphics mode.

REVERSEY

reverses the order in which the *y* pages are drawn. By default, the pages are ordered from bottom to top in the graphics mode. This option orders them from top to bottom.

ROTATE

rotates the network diagram to change the orientation of the network to be from top to bottom instead of from left to right. For example, you can use this option to draw a Bill of Materials diagram that is traditionally drawn from top to bottom with the Final Product drawn at the top of the tree. In addition to rotating the orientation of the network, use the ROTATETEXT option to rotate the text within each node. See Example 9.18 for an illustration of this option.

This option is similar to the global graphics option, ROTATE (GOPTIONS ROTATE). Note that if the global graphics option is used, titles and footnotes also need to be drawn with an angle specification:

A=90. However, some device drivers ignore the global graphics option, ROTATE (for example, the SASGDDMX driver). Use the ROTATE option on the ACTNET statement for such device drivers.

ROTATETEXT

RTEXT

rotates the text within the nodes by 90 degrees. This option is useful when used in conjunction with the ROTATE option in the ACTNET statement (or the global graphics option ROTATE) to change the orientation of the network to be from top to bottom instead of from left to right. For example, you can use this option to draw an organizational chart that is traditionally drawn from top to bottom with the head of the organization at the top of the chart. If the ROTATETEXT option is specified, then the definitions of the BOXHT= and BOXWIDTH= options are reversed. See Example 9.18 for an illustration of this option.

SEPARATEARCS

separates the arcs to follow distinct tracks. By default, the procedure draws all segments of the arcs along a central track between the nodes, which may cause several arcs to be drawn on top of one another. If the SEPARATEARCS option is specified, the procedure may increase the values of the XBETWEEN= and YBETWEEN= options to accommodate the required number of lines between the nodes.

VMARGIN=*integer*

specifies the width of a vertical margin (in number of character cell positions) for the network. The default width is 1.

VPAGES=*v*

NYPAGES=*v*

specifies that the network diagram is to be produced using *v* vertical pages. This, however, may not be possible due to intrinsic constraints on the output. For example, PROC NETDRAW requires that every vertical page should contain at least one *y* level. Thus, the number of vertical pages can never exceed the number of horizontal levels in the network. The exact number of vertical pages used by the procedure is provided in the _ORNETDR macro variable. See the section "Macro Variable _ORNETDR" on page 704 for further details.

The appearance of the diagram with respect to the VPAGES= option is also influenced by the presence of other related procedure options. The VPAGES= option performs the task of determining the number of horizontal pages in the absence of the HPAGES= option (or the NXNODES= option). If the COMPRESS or PCOMPRESS option is specified (without the HPAGES= or NXNODES= options), the chart uses one horizontal page. If neither the COMPRESS nor PCOMPRESS option is specified, the number of horizontal pages is computed in order to display as much of the chart as possible in a proportional manner.

WEB=*variable*

HTML=*variable*

specifies the character variable in the Network data set that identifies an HTML page for each activity. The procedure generates an HTML image map using this information for each node in the network diagram.

ZONEPAT

indicates that if a PATTERN= variable is not specified or is missing and if a ZONE= variable is present, then the node patterns are based on the value of the ZONE= variable.

Line-Printer Options

BRKCHAR=*brkchar*

specifies the character used for drawing the zigzag break lines down the chart at break points of the time axis. The default value is >. This option is valid only for time-scaled network diagrams.

FORMCHAR [*index list***]=***'string'*

specifies the characters used for node outlines and arcs. The value is a string 20 characters long. The first 11 characters define the 2 bar characters, vertical and horizontal, and the 9 corner characters: upper-left, upper-middle, upper-right, middle-left, middle-middle (cross), middle-right, lower-left, lower-middle, and lower-right. These characters are used to outline each node and connect the arcs. The nineteenth character denotes a right arrow. The default value of the FORMCHAR= option is `|----|+|---+=|-/\<>*`. Any character or hexadecimal string can be substituted to customize the appearance of the diagram. Use an index list to specify which default form character each supplied character replaces, or replace the entire default string by specifying the full character replacement string without an index list. For example, change the four corners of each node and all turning points of the arcs to asterisks by specifying

```
FORMCHAR(3 5 7 9 11)= '*****'
```

Specifying

```
formchar='           ' (11 blanks)
```

produces a network diagram with no outlines for the nodes (as well as no arcs). For further details about the FORMCHAR= option see Chapter 7, "The DTREE Procedure," and Chapter 8, "The GANTT Procedure."

REFCHAR=*refchar*

specifies the reference character used for drawing reference lines. The default value is "|". This option is valid only for time-scaled network diagrams.

Details: NETDRAW Procedure

Network Input Data Set

The Network input data set contains the precedence information, namely the activity-successor information for all the nodes in the network. The minimum amount of information that is required by PROC NETDRAW is the activity-successor information for the network. Additional information in the input data set can be used by the procedure to add detail to the nodes in the diagram or control the layout of the network diagram.

Three types of data sets are typically used as the Network data set input to PROC NETDRAW. Which type of data set you use depends on the stage of the project:

- The Activity data set that is input to PROC CPM is the first type. In the initial stages of project definition, it may be useful to get a graphical representation of the project showing all the activity precedence constraints.
- The Schedule data set produced by PROC CPM (as the OUT= data set) is the second type. When a project is in progress, you may want to obtain a network diagram showing all the relevant start and finish dates for the activities in the project, in addition to the precedence constraints. You may also want to draw a time-scaled network diagram, with the activities arranged according to the start or finish times corresponding to any of the different schedules produced by PROC CPM.
- The Layout data set produced by PROC NETDRAW (as the OUT= data set) is the third type. Often, you may want to draw network diagrams of the project every week showing updated information (as the project progresses); if the network logic has not changed, it is not necessary to determine the placement of the nodes and the routing of the arcs every time. You can use the Layout data set produced by PROC NETDRAW that contains the node and arc positions, update the start and finish times of the activities or merge in additional information about each activity, and use the modified data set as the Network data set input to PROC NETDRAW. The new network diagram will have the same layout as the earlier diagram but will contain updated information about the schedule. Such a data set may also be useful if you want to modify the layout of the network by changing the positions of some of the nodes. See the section "Controlling the Layout" on page 695 for details about how the layout information is used by PROC NETDRAW. If the Layout data set is used, it contains the variables _FROM_ and _TO_; hence, it is not necessary to specify the ACTIVITY= and SUCCESSOR= options. See Example 9.13 and Example 9.14 for illustrations of the use of the Layout data set.

The minimum information required by PROC NETDRAW from the Network data set is the variable identifying each node in the network and the variable (or variables) identifying the immediate successors of each node. In addition, the procedure can use other optional variables in the data set to enhance the network diagram. The procedure uses the variables specified in the ID= option to label each node. The procedure also looks for default variable names in the Network data set that are added to the list of ID variables; the default variable names are E_START, E_FINISH, L_START, L_FINISH, S_START, S_FINISH, A_START, A_FINISH, T_FLOAT, and F_FLOAT. The format used for determining the location of these variables within each node is described in the section "Format of the Display" on page 692. See the section "Variables in the Network Data Set" on page 689 for a table of all the variables in the Network data set and their interpretations by PROC NETDRAW.

If the Network data set contains the variables _X_ and _Y_ identifying the *x* and *y* coordinates of each node and each turning point of each arc in the network, then this information is used by the procedure to draw the network. Otherwise, the precedence relationships among the activities are used to determine the layout of the network. It is possible to specify only the node positions and let the procedure determine the routing of all the arcs. However, partial information cannot be augmented by the procedure.

NOTE: If arc information is provided, the procedure assumes that it is complete and correct and uses it exactly as specified.

Variables in the Network Data Set

The NETDRAW procedure expects all the network information to be contained in the Network input data set named by the DATA= option. The network information is contained in the ACTIVITY and SUCCESSOR variables. In addition, the procedure uses default variable names in the Network data set for specific purposes. For example, the _X_ and _Y_ variables, if they are present in the Network data set, represent the coordinates of the nodes, the _SEQ_ variable indexes the turning points of each arc of the network, and so on.

In addition to the network precedence information, the Network data set may also contain other variables that can be used to change the default layout of the network. For example, the nodes of the network can be aligned in the horizontal direction using the ALIGN= specification, or they can be divided into horizontal bands (or zones) using a ZONE variable.

Table 9.2 lists all of the variables associated with the Network data set and their interpretations by the NETDRAW procedure. Note that all the variables are identified to the procedure in the ACTNET statement. Some of the variables use default names that are recognized by the procedure to denote specific information, as explained previously. The table indicates if the variable is default or needs to be identified in the ACTNET statement.

Table 9.2 Network Data Set and Associated Variables

Statement	Variable Name	Interpretation
ACTNET	ACTIVITY	Activity or node name
	ALIGN	Align variable for time-scaled network
	DURATION	Duration of activity
	ID	Additional variables to be displayed
	PATTERN	Pattern number
	SUCCESSOR	Immediate successor
	WEB	HTML page corresponding to activity
	ZONE	Zone variable for dividing network
Default Variable Names	A_FINISH	Default ID variable
	A_START	Default ID variable
	E_FINISH	Default ID variable
	E_START	Default ID variable
	F_FLOAT	Default ID variable
	L_FINISH	Default ID variable
	L_START	Default ID variable
	S_FINISH	Default ID variable
	S_START	Default ID variable
	T_FLOAT	Default ID variable
	FROM	Supersedes ACTIVITY= specification
	_PATTERN	Supersedes PATTERN= specification
	SEQ	Index of turning point in arc
	_TO_	Supersedes SUCCESSOR= specification
	_X_	*x* coordinate of node or arc turning point
	_Y_	*y* coordinate of node or arc turning point

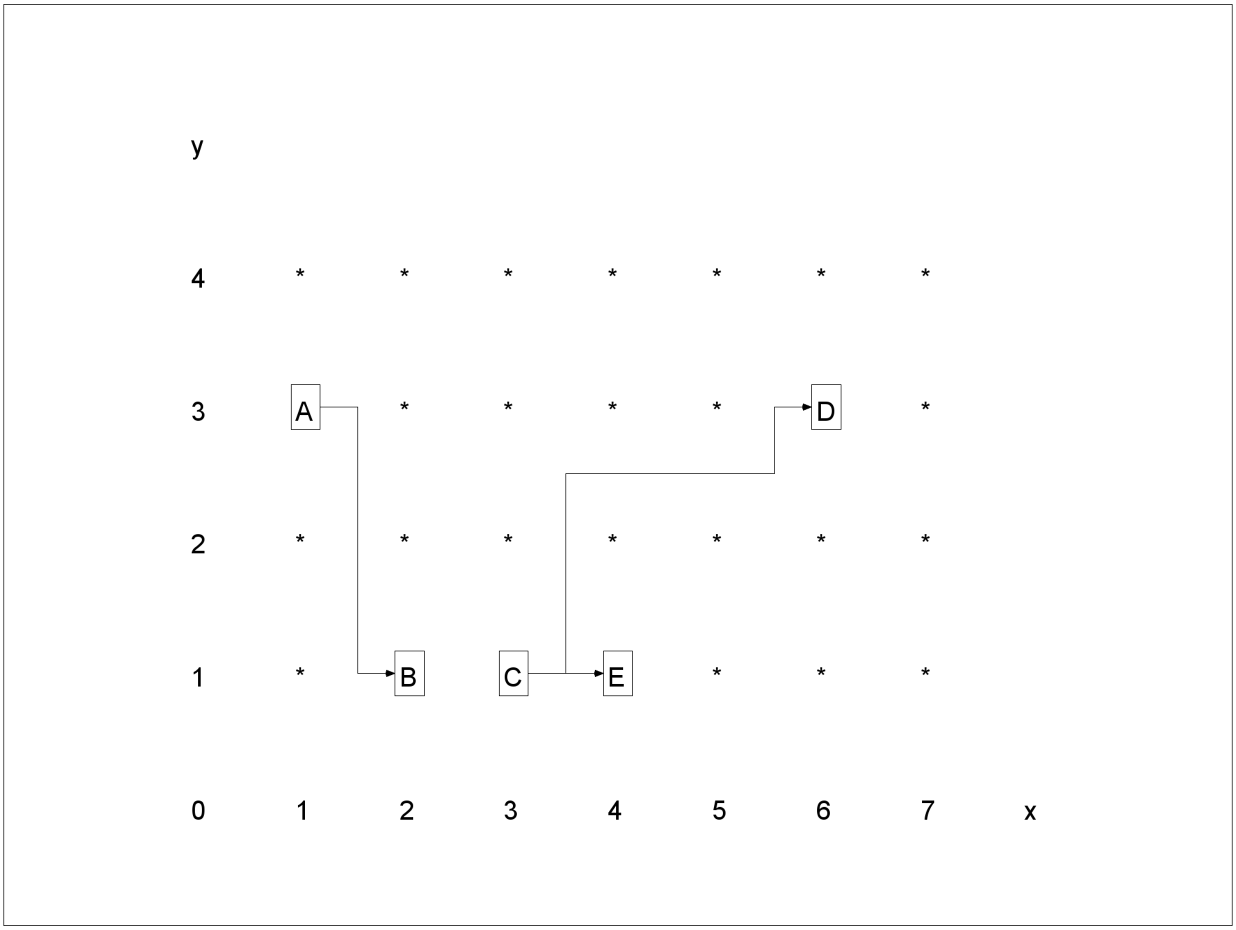
Missing Values

Missing values are not allowed for the ACTIVITY, _X_, _Y_, and _SEQ_ variables. Missing values for the SUCCESSOR and ID variables are ignored. Missing values are not allowed for the ALIGN= variable if the QUITMISSINGALIGN option is specified; otherwise, the procedure determines suitable values for the ALIGN= variable using the topological ordering of the network nodes.

Layout of the Network

The network layout is determined in two stages. First, the precedence relationships are used to determine the positions of the nodes, which are then used to determine a routing of the arcs. The positions of the nodes and arcs are identified by specifying their x and y coordinates in a grid. Figure 9.7 shows a sample grid and explains some of the conventions followed by PROC NETDRAW in determining the node and arc layout. This notation will be useful in later sections that describe the Layout data set and how you can control the layout of the diagram. The asterisks in the figure represent possible positions for the nodes of the network. The arcs are routed between the possible node positions. For example, node **A** has coordinates $(1, 3)$ and node **B** has coordinates $(2, 1)$. The arc connecting them has two turning points and is completely determined by the two pairs of coordinates $(1.5, 3)$ and $(1.5, 1)$; here, $x = 1.5$ implies that the position is midway between the x coordinates *1* and *2*.

Figure 9.7 Sample Grid and Coordinates for Node and Arc Layout



PROC NETDRAW sets $x = 1$ for all nodes with no predecessors; the x coordinates for the other nodes are determined so that each node is placed to the immediate right of all its predecessors; in other words, no node will appear to the left of any of its predecessors or to the right of any of its successors in the network diagram. The nodes are placed in topological order: a node is placed only after all its predecessors have been placed. Thus, the node-placement algorithm requires that there should be no cycles in the network. The y coordinates of the nodes are determined by the procedure using several heuristics designed to produce a reasonable compact diagram of the network. To draw a network that has cycles, use the BREAKCYCLE option, or you can specify the node coordinates or an ALIGN= variable to circumvent the requirement of a topological ordering of the nodes (see the second part of Example 9.12).

Note that the x and y coordinates fix only a relative positioning of the nodes and arcs. The actual distance between two nodes, the width and height of each node, and so on can be controlled by specifying desired values for the options that control the format of the display, namely, BOXHT=, BOXWIDTH=, and so on. See the section "Format of the Display" on page 692 for details about these options.

By default, the procedure routes the arcs using a simple heuristic that uses, at most, four turning points: the arc leaves the predecessor node from its right edge, turns up or down according to whether the successor is above or below the current node position, then tracks horizontally across to the vertical corridor just before the successor node, and then tracks in a vertical direction to meet the successor node. For example, see the tracking of the arc connecting nodes **C** and **D** in Figure 9.7.

For networks that include some nonstandard precedence constraints, the arcs may be drawn from and to the appropriate edges of the nodes, depending on the type of the constraint.

The default routing of the arcs may lead to an unbalanced diagram with too many arcs in one section and too few in another. The DP option in the ACTNET statement causes the procedure to use a dynamic programming algorithm to route the arcs. This algorithm tries to route the arcs between the nodes so that not too many arcs pass through any interval between two nodes. The procedure sets the maximum number of arcs that are allowed to be routed along any corridor to be equal to the maximum number of successors for any node. The HTRACKS= and VTRACKS=options enable you to set these maximum values: HTRACKS specifies the maximum number of arcs that are allowed to pass horizontally through any point while VTRACKS specifies the same for arcs in the vertical direction. See Example 9.7 for an illustration of the HTRACKS= option.

The layout of the network for time-scaled and zoned network diagrams is discussed in the section "Time-Scaled Network Diagrams" on page 696 and the section "Zoned Network Diagrams" on page 698, respectively. the section "Organizational Charts or Tree Diagrams" on page 698 describes the layout of the diagram when the TREE option is specified.

Format of the Display

As explained in the previous section, the layout of the network is determined by the procedure in terms of *x* and *y* coordinates on a grid as shown in Figure 9.7. The distance between nodes and the width and height of each node is determined by the values of the format control options: XBETWEEN=, YBETWEEN=, BOXHT=, and BOXWIDTH= . Note that if the ROTATETEXT option is specified (in graphics mode), then the definitions of the BOXHT= and BOXWIDTH= options are reversed.

The amount of information that is displayed within each node is determined by the variables specified by the ID= option, the number of default variables found in the Network data set, and whether the NOLABEL and NODEFID options are specified. The values of the variables specified by the ID= option are placed within each node on separate lines. If the NOLABEL option is in effect, only the values of the variables are written; otherwise, each value is preceded by the name of the ID variable truncated to three characters. Recall from the section "Syntax: NETDRAW Procedure" on page 670 that, in addition to the variables specified using the ID= option, the procedure also displays additional variables. These variables are displayed below the variables explicitly specified by the ID= option, in pre-determined relative positions within each node (see Table 9.3.)

Table 9.3 Display Format for the Variables within Each Node

ID1	
.	
.	
.	
ID*n*	
Activity *variable*	Duration *variable*
E_START	E_FINISH
L_START	L_FINISH
S_START	S_FINISH
A_START	A_FINISH
T_FLOAT	F_FLOAT

NOTE: If a node is identified as a successor (through a SUCCESSOR variable) and is never identified with the ACTIVITY variable, the ID values for this node are never defined in any observation; hence, this node will have missing values for all the ID variables.

If the SHOWSTATUS option is specified and the Network data set contains progress information (in either the STATUS variable or the A_START and A_FINISH variables), the procedure appropriately marks each node referring to activities that are completed or in progress. See Example 9.8 for an illustration of the SHOWSTATUS option.

The features just described pertain to all three modes of the procedure. In addition, there are options to control the format of the display that are specific to the mode of invocation of the procedure. For graphics quality network diagrams, you can choose the color and pattern used for each node separately by specifying a different pattern number for the PATTERN variable, identified in the ACTNET statement (for details, see the section "Graphics Version" on page 702). For line-printer or full-screen network diagrams, the FORMCHAR= option enables you to specify special boxing characters that enhance the display; for full-screen network diagrams, you can also choose the color of the nodes using the PATTERN= option.

By default, all arcs are drawn along the center track between two consecutive nodes. The SEPARATEARCS option, which is available in the graphics version, separates arcs in the same corridor by drawing them along separate tracks, thus preventing them from being drawn on top of each other.

If the network fits on one page, it is centered on the page; in the graphics mode, you can use the NOVCENTER option to prevent centering in the vertical direction so that the network is drawn immediately below the title. If the network cannot fit on one page, it is split onto different pages appropriately. See the section "Page Format" on page 693 for a description of how the pages are split.

Page Format

Figure 9.8 Page Layout

7	8	9
4	5	6
1	2	3

As explained in the section "Format of the Display" on page 692, if the network fits on one page, it is centered on the page (unless the NOVCENTER option is specified); otherwise, it is split onto different pages appropriately, and each page is drawn starting at the bottom left corner. If the network is drawn on multiple pages, the procedure numbers each page of the diagram on the top right corner of the page. The pages are numbered starting with the bottom left corner of the entire picture. Thus, if the network diagram is broken into three horizontal and three vertical levels and you want to paste all the pieces together to form one picture, they should be arranged as shown in Figure 9.8.

The number of pages of graphical output produced by the NETDRAW procedure depends on several options such as the NXNODES=, NYNODES=, HPAGES=, VPAGES=, COMPRESS, PCOMPRESS, HEIGHT=, and the ID= options. The value of the HTEXT= option and the number of variables specified in the ID= options determines the size of each node in the network diagram, which in turn affects the number of horizontal and vertical pages needed to draw the entire network. The number of pages is also affected by the global specification of the HPOS=, VPOS=, HSIZE=, and VSIZE= graphics options.

The COMPRESS and PCOMPRESS options force the entire network diagram to be drawn on a single page. You can explicitly control the number of horizontal and vertical pages using the HPAGES= and VPAGES= options. The NXNODES= and NYNODES= options enable you to specify the number of nodes in the horizontal and vertical directions, respectively, on each page of the network diagram.

For examples of these options and how they affect the network diagram output, see Example 9.5.

Layout Data Set

The Layout data set produced by PROC NETDRAW contains all the information needed to redraw the network diagram for the given network data. In other words, the Layout data set contains the precedence information, the ID variables that are used in the current invocation of the procedure, and variables that contain the coordinate information for all the nodes and arcs in the network.

The precedence information used by the procedure is defined by two new variables named _FROM_ and _TO_, which replicate the ACTIVITY and SUCCESSOR variables from the Network data set. Note that the Layout data set has only one _TO_ variable even if the Network data set has multiple SUCCESSOR variables; if a given observation in the Network data set defines multiple successors for a given activity, the Layout data set defines a new observation for each of the successors. In fact, for each (node, successor) pair, a sequence of observations, defining the turning points of the arc, is saved in the Layout data set; the number of observations corresponding to each pair is equal to one plus the number of turns in the arc connecting the node to its successor. Suppose that a node 'C' has two successors, 'D' and 'E,' and the arcs connecting 'C' and 'D' and 'C' and 'E' are routed as shown in Figure 9.7. Then, Table 9.4 illustrates the format of the observations corresponding to the two (_FROM_, _TO_) pairs of nodes, ('C', 'D') and ('C', 'E').

Table 9.4 Sample Observations in the Layout Data Set

FROM	_TO_	_X_	_Y_	_SEQ_	_PATTERN	ID variables
C	D	3	1	0	1	
C	D	3.5	1	1	.	
C	D	3.5	2.5	2	.	
C	D	5.5	2.5	3	.	
C	D	5.5	3	4	.	
C	E	3	1	0	1	
		.			.	
		.			.	
		.			.	

For every (node, successor) pair, the first observation (_SEQ_ = '0') gives the coordinates of the predecessor node; the succeeding observations contain the coordinates of the turning points of the arc connecting the predecessor node to the successor. The data set also contains a variable called _PATTERN, which contains the pattern number that is used for coloring the node identified by the _FROM_ variable. The value of this variable is missing for observations with _SEQ_ > 0.

Controlling the Layout

As explained in the section "Layout of the Network" on page 690, the procedure uses the precedence constraints between the activities to draw a reasonable diagram of the network. A very desirable feature in any procedure of this nature is the ability to change the default layout. PROC NETDRAW provides two ways of modifying the network diagram:

- using the full-screen interface
- using the Network data set

The full-screen method is useful for manipulating the layout of small networks, especially networks that fit on a handful of screens. You can use the full-screen mode to examine the default layout of the network and move the nodes to desired locations using the MOVE command from the command line or by using the appropriate function key. When a node is moved, the procedure reroutes all the arcs that connect to or from the node; other arcs are unchanged. For details about the MOVE command, see the section "Full-Screen Version" on page 699.

You can use the Network data set to modify or specify completely the layout of the network. This method is useful if you want to draw the network using information about the network layout that has been saved from an earlier invocation of the procedure. Sometimes you may want to specify only the positions of the node and let the procedure determine the routing of the arcs. The procedure looks for three default variables in the data set: _X_, _Y_, and _SEQ_ . The _X_ and _Y_ variables are assumed to denote the *x* and *y* coordinates of the nodes and all the turning points of the arcs connecting the nodes. The variable _SEQ_ is assumed to denote the order of the turning points. This interpretation is consistent with the values assigned to the _X_, _Y_, and _SEQ_ variables in the Layout data set produced by PROC NETDRAW. If there is no variable called _SEQ_ in the data set, the procedure assumes that only the node positions are specified and uses the specified coordinates to place the nodes and determines the routing of the arcs corresponding to these positions. If there is a variable called _SEQ_, the procedure requires that the turning points for each arc be specified in the proper order, with the variable _SEQ_ containing numbers sequentially starting with 1 and continuing onward. The procedure then draws the arcs exactly as specified, without checking for consistency or interpolating or extrapolating turning points that may be missing.

The ALIGN= variable provides another means of controlling the node layout (see the section "Time-Scaled Network Diagrams" on page 696). This variable can be used to specify the *x* coordinates for the different nodes of the network; the procedure then determines the *y* coordinates. Note that time-scaled network diagrams (without an ALIGN= specification) are equivalent to network diagrams drawn with the ALIGN= variable being set to the E_START variable.

You can also control the placement of the nodes using the ZONE=option (see the section "Zoned Network Diagrams" on page 698). The procedure uses the values of the ZONE variable to divide the network into horizontal zones. Thus, you can control the horizontal placement of the nodes using the ALIGN= option and the vertical placement of the nodes using the ZONE= option.

For networks that have a tree structure, the TREE option draws the network as a tree, thus providing another layout option (see the section "Organizational Charts or Tree Diagrams" on page 698). The procedure draws the tree from left to right, with the root at the left edge of the diagram. Thus, the children of each node are drawn to the right of the node. In the graphics mode of invocation, you can use the ROTATETEXT option in conjunction with the ROTATE option in the ACTNET statement (or the global graphics option ROTATE) to obtain a top-down tree diagram.

Time-Scaled Network Diagrams

By default, PROC NETDRAW uses the topological ordering of the activity network to determine the *x* coordinates of the nodes. As a project progresses, you may want to display the activities arranged according to their time of occurrence. Using the TIMESCALE option, you can draw the network with a time axis at the top and the nodes aligned according to their early start times, by default. You can use the ALIGN= option to specify any of the other start or finish times in the Network data set. In fact, PROC NETDRAW enables you to align the nodes according to any numeric variable in the data set.

If the TIMESCALE option is specified without any ALIGN= specification, the procedure chooses one of the following variables as the ALIGN= variable: E_START, L_START, S_START, or A_START, in that order. The first of these variables that is found is used to align the nodes. The minimum and maximum values of the ALIGN= variable are used to determine the time axis. The format of this variable is used to determine the default value for the MININTERVAL= option. The value of the MININTERVAL= option (or the default value) is used to determine the format of the time axis. You can override the format based on *mininterval* by specifying the desired format for the ALIGN= variable (using the FORMAT statement to indicate a standard SAS format or a special user-defined format) and the USEFORMAT option in the ACTNET statement. Table 9.5 lists the valid values of *mininterval* corresponding to the type of the ALIGN= variable and the default format corresponding to each value of *mininterval*. For each value in the first column, the first value of *mininterval* listed is the default value of the MININTERVAL= option corresponding to that type of the ALIGN= variable.

Several options are available in PROC NETDRAW to control the spacing of the nodes and the scaling of a time-scaled network diagram:

- The MININTERVAL= option enables you to scale the network diagram: one tick mark is associated with one unit of *mininterval*. Thus, if *mininterval* is DAY, each column is used to represent one day and all activities that start on the same day are placed in the same column. By default, the procedure omits any column (tick mark) that does not contain any node.
- The LINEAR option enables you to print a tick mark corresponding to every day (or the unit of *mininterval*). Note that, for a project that has few activities spread over a large period of time, the LINEAR option can lead to a network diagram that is very wide.
- The MAXNULLCOLUMN= option specifies the maximum number of empty columns that is allowed between two consecutive nonempty columns. The LINEAR option is equivalent to specifying *maxncol* = infinity, while the default time-scaled network diagram is drawn with *maxncol* = 0.

- The NLEVELSPERCOLUMN= option enables you to contract the network diagram by combining a few columns. For example, if *mininterval* is DAY and *nlevelspercol* is 7, each column contains activities that start within seven days of each other; note that the same effect can be achieved by setting *mininterval* to be WEEK.

Table 9.5 MININTERVAL Values and Axis Format

ALIGN Variable Type	MININTERVAL	Axis Label Format
number		numeric format
SAS time	HOUR	HHMM5.
	MINUTE	HHMM5.
	SECOND	TIME8.
SAS date	DAY	DATE7.
	WEEKDAY	DATE7.
	WEEK	DATE7.
	MONTH	MONYY5.
	QTR	MONYY5.
	YEAR	MONYY5.
SAS datetime	DTDAY	DATE7.
	WORKDAY	DATE7.
	DTWRKDAY	DATE7.
	DTSECOND	DATETIME16.
	DTMINUTE	DATETIME16.
	DTHOUR	DATETIME13.
	DTWEEK	DATE7.
	DTMONTH	MONYY5.
	DTQTR	MONYY5.
	DTYEAR	MONYY5.

The node-placement algorithm described in the section "Layout of the Network" on page 690 is modified slightly for time-scaled network diagrams. The *x* coordinate of each node is determined by the value of the ALIGN= variable. The scaling options just described are used to determine the tick mark corresponding to the node. The *y* coordinate is determined as before. Once the node placement is completed, the arc routing algorithm is the same as described earlier.

NOTE: Since the node placement for time-scaled networks is determined by the ALIGN= variable, it is possible that some of the arcs between the nodes may have to be routed from right to left instead of from left to right; in other words, there may be some backward arcs. Note also that, if the ALIGN= variable is used to determine the *x* coordinates of the nodes, the procedure can also draw networks that contain cycles (see the second part of Example 9.12).

Several other options are available to control the appearance of time-scaled network diagrams: AUTOREF, BRKCHAR=, CAXIS=, CREF=, CREFBRK=, FRAME, LREF=, LREFBRK=, NOREPEATAXIS, NOTIMEAXIS, REFBREAK, REFCHAR=, and SHOWBREAK. These options are described in the section "Syntax: NETDRAW Procedure" on page 670.

Zoned Network Diagrams

Most projects have at least one natural classification of the different activities in the project: department, type of work involved, location of the activity, and so on. The ZONE= option enables you to divide the network diagram into horizontal bands or zones corresponding to this classification. The procedure uses the following rules to place the nodes in a zoned network diagram:

- The values of the ZONE variable are used to define as many zones as there are distinct values of this variable.
- Each node of the network is drawn within its corresponding zone.
- The number of rows within each zone is determined by the maximum number of nodes in any given column that correspond to that zone.
- The values of the ZONE variable do not need to be sorted in any particular order, nor do they need to be grouped by distinct values.
- The zones are ordered according to the order of appearance of the different values of the ZONE variable in the Network data set. This enables you to choose any order for the zone values.
- For arcs that connect two nodes within the same zone, the arc lies entirely within the zone; in other words, all the turning points of the arc have *y* coordinates that are between the minimum and maximum *y* coordinates for the zone.
- Each zone is labeled by the value of the ZONE variable unless the NOZONELABEL option is specified.
- Each zone is separated from the next by a horizontal line drawn across the width of the network unless the NOZONELABEL option is specified.
- In the graphics and full-screen modes of invocation of the procedure, you can use the ZONEPAT option to color the nodes in each zone differently using different pattern statements. In the graphics mode, the first zone uses the first PATTERN statement, the second zone uses the second PATTERN statement, and so on; in full-screen mode, the colors available for the device are repeated in cyclic order. Note that the values of the PATTERN variable (or the default _PATTERN variable, if it exists in the Network data set) override the node patterns dictated by the ZONEPAT option.

Organizational Charts or Tree Diagrams

The NETDRAW procedure automatically draws any acyclic network; it does not have to be a representation of a project. You can also use the procedure to draw a general directed graph that has cycles, if node location is specified or if the BREAKCYCLE option is specified. The procedure attempts to draw the network in a compact fashion, which may not always produce the expected result. Trees form one such class of directed graphs that have an inherent natural layout that may not be produced by the default layout of PROC NETDRAW. The TREE option in the ACTNET statement exploits the tree structure of the network by laying the nodes out in the form of a tree.

A directed graph is said to be a tree if it has a root and there is a unique directed path from the root to every node in the tree. An equivalent characterization of a tree is that the root node has no predecessors and every other node has exactly one predecessor (Even 1979). Typical examples of trees that arise in project management are organizational charts or work breakdown structures. If the TREE option is specified, the NETDRAW procedure checks if the network has a tree structure and draws the network with the root at the left edge of the diagram and the children of each node appearing to the right of the node. In other words, the tree is drawn from left to right.

The NETDRAW procedure enables you to specify multiple trees in the same Network data set; each tree is drawn separately in the same diagram with all the roots appearing at the left edge of the diagram. Thus, you can use the TREE option as long as every node in the network has *at most* one predecessor. It you specify the TREE option and some node has multiple predecessors, the TREE option is ignored and the procedure uses the default node-layout algorithm.

There are several features that control the appearance of the tree:

- The children of each node are placed in the order of occurrence in the Network data set. The (x, y) coordinates of each node are required to be integers. The procedure attempts to place each node at the center of all its children, subject to the requirement that the coordinates must be integers. This requirement may cause some of the nodes to be positioned slightly off-center. See Example 9.15.

- The SEPARATESONS option separates the children of a node, if necessary, to enable the parent node to be exactly centered with respect to its children. See the second part of Example 9.15.

- The CENTERSUBTREE option can be used to center each node with respect to the entire subtree originating from the node instead of centering it with respect to its children.

- In graphics mode, you can change the orientation of the network to be from top to bottom instead of from left to right. To do so, use the ROTATETEXT option in the ACTNET statement to rotate the text within the nodes and the ROTATE option in the ACTNET statement (or the ROTATE global graphics option) to rotate the entire diagram by 90 degrees. See Example 9.18 for an illustration of this feature.

Full-Screen Version

You can invoke PROC NETDRAW in full-screen mode by specifying FS (or FULLSCREEN) in the PROC NETDRAW statement. The statement specifications are the same as for the line-printer mode. The full-screen mode offers you a convenient way to browse the network diagram of the project and change the layout of the network by moving the nodes of the network to desired locations. However, you cannot move a node to any position that violates the precedence constraints that must be satisfied by the node. In other words, you cannot move a node to the left of any of its predecessors or to the right of any of its successors. For time-scaled network diagrams, you cannot move a node out of the column corresponding to the value of the ALIGN= variable. For zoned network diagrams you cannot move a node out of its zone.

The format control options are treated in the same way as for the line-printer version, with some minor changes. It is assumed that the main purpose of invoking the procedure is to gain a general picture of the layout of the entire network and to modify it to some extent. In an effort to display as much of the network as possible, the initial display on the screen is drawn with only one row and three columns for each node. In other words, the BOXHT=, BOXWIDTH=, XBETWEEN=, and YBETWEEN= options are ignored by the procedure in drawing the initial display. However, the full-screen commands supported by PROC NETDRAW enable you to change the scale of the diagram. You can display as much or as little information within each node by invoking the SCALE ROW or the SCALE COL command or both. The SCALE MAX command causes the procedure to display the diagram using the values specified in the ACTNET statement or the dimensions that would be required to display all the ID information, whichever is larger. The SCALE RESET command returns the scaling to the initial values used for display.

The nodes of the network are color coded on the basis of the PATTERN variable. If there is no PATTERN variable, then the nodes are color coded depending on whether the activities are normal, critical, or supercritical. The nodes are drawn in reverse video. By default, the nodes are drawn without an outline; however, there is an OUTLINE command that lets you toggle back and forth between an outlined or non-outlined node. Using an outline for the node is useful if you want to obtain a printout of the screen display using SPRINT; it helps mark the boundary of each node clearly.

Commands

Table 9.6 lists the commands that can be invoked from the command line in the full-screen version of PROC NETDRAW. These commands are explained in greater detail in this section.

Table 9.6 Full-Screen Commands and Their Purposes

Scrolling	Controlling Display	Changing Network Layout	Exiting
BACKWARD	OUTLINE	CLEAR	GEND
FORWARD	SCALE	MOVE	END
LEFT			CANCEL
RIGHT			
TOP			
BOTTOM			
VSCROLL			
HSCROLL			

BACKWARD
: scrolls toward the top of the network by the VSCROLL amount. BACKWARD MAX scrolls to the top of the network. You can specify the vertical scroll amount for the current command as BACKWARD PAGE | HALF | *n*.

BOTTOM
: scrolls to the bottom of the network.

CANCEL
: ends the current invocation of the procedure.

CLEAR
: clears any outstanding move commands.

GEND
ends the current invocation of the procedure after drawing the network in graphics mode with the compress option.

END
ends the current invocation of the procedure.

FORWARD
scrolls toward the bottom of the network by the VSCROLL amount. FORWARD MAX scrolls to the bottom of the network. You can also specify the vertical scroll amount for the current command as FORWARD PAGE | HALF | *n*.

HELP
displays a help screen listing all the full-screen commands specific to PROC NETDRAW.

HOME
moves the cursor to the command line.

HSCROLL
sets the amount that information scrolls horizontally when you execute the LEFT or RIGHT command. The format is HSCROLL PAGE | HALF | *n*. The specification is assumed to be in number of horizontal levels. HSCROLL PAGE sets the scroll amount to be the number of horizontal levels that fit on one screen; HSCROLL HALF is half that amount; HSCROLL *n* sets the horizontal scroll amount to *n* levels.

KEYS
displays current function key settings for the NETDRAW procedure.

LEFT
scrolls toward the left boundary of the network by the HSCROLL amount. LEFT MAX scrolls to the left boundary. You can specify the horizontal scroll amount for the current command as LEFT PAGE | HALF | *n*.

MOVE
specifies a node to be moved or a place to move a node to. You can specify these in any order. Thus, you can first position the cursor on the node that you want to move, issue the MOVE command, and then position the cursor at a target position and issue the MOVE command again. If the target position is valid, the node is moved. You can also first specify the target position and then indicate the node that is to be moved.

NOTE: For a standard network, a node cannot be moved to any position that violates the topological ordering of the nodes in the network. For time-scaled network diagrams, you cannot move a node to a level corresponding to a different tick mark. For zoned network diagrams, you cannot move a node out of its zone.

OUTLINE
causes an outline to be drawn around each node in the network. This is useful if you want to print a copy of the screen by using the SPRINT command. The OUTLINE command works like an on/off switch: you can turn it off by entering the command again.

RIGHT

scrolls toward the right boundary of the network by the HSCROLL amount. RIGHT MAX scrolls to the right boundary. You can also specify the horizontal scroll amount for the current command as RIGHT PAGE | HALF | *n*.

SCALE

controls the scaling of the nodes and the space between nodes. The format of this command is SCALE MAX | MIN | RESET | ROW MAX | COL MAX | ROW MIN | COL MIN | ROW *n* | COL *n* | +*n* | -*n*. The number *n* denotes the number of character positions. SCALE MIN displays as many nodes on the screen as can fit. SCALE MAX enables as many rows and columns per node as is required to display all the information that pertains to it. SCALE ROW MAX displays the maximum number of rows per node. SCALE COL MAX displays the maximum number of columns per node. SCALE ROW *n* sets the number of rows per node to *n*. SCALE ROW $+n$ increases the number of rows per node by *n*. SCALE COL *n* sets the number of columns per node to *n*. SCALE COL $+n$ increases the number of columns per node by *n*. SCALE RESET sets the values to be the same as for the initial display. Note that none of these values can be greater than the dimensions of the screen.

TOP

scrolls to the top of the network.

VSCROLL

sets the amount by which information scrolls vertically when you execute the BACKWARD or FORWARD command. The format is VSCROLL PAGE | HALF | *n*. The specification is assumed to be in number of vertical levels. VSCROLL PAGE sets the scroll amount to be the number of vertical levels that fit on one screen; VSCROLL HALF is half that amount; VSCROLL *n* sets the vertical scroll amount to *n* levels.

Full-Screen Global Commands

Most of the global commands used in SAS/FSP software are also valid with PROC NETDRAW. Some of the commands used for printing screens are described in the section “Global Commands” on page 535 in Chapter 8, “The GANTT Procedure.”

Graphics Version

Several options are available in the ACTNET statement to enhance the appearance of the network diagram in graphics mode. These are described in the section “Graphics Options” on page 681. The format control options BOXWIDTH=, BOXHT=, XBETWEEN=, and YBETWEEN= are also valid in this mode and can be used to control the width and height of each node and the distance between the nodes. These parameters are specified in terms of number of character cell positions. The number of positions available on one page depends on the graphics device that is used; thus, if a plotter is used with large paper, more of the network will be drawn on a single page. Further, you can control the number of character cell positions on a page by changing the values of the global graphics options (HPOS= and VPOS=). Note that the NETDRAW procedure is not supported with the ActiveX or Java series of devices on the GOPTIONS statement.

You can also control the number of nodes on a given page by specifying the NXNODES= and NYNODES= options. The HPAGES= and VPAGES= options control the number of pages in the horizontal and vertical directions. Thus, you have a wide degree of control over the amount of information displayed on each page of the network diagram.

Another option that is available in graphics mode to control the appearance of your network diagrams is the specification of a PATTERN variable in the ACTNET statement. If the variable is named _PATTERN, you do not need to use the PATTERN= option; the procedure looks for such a variable by default. You can use this variable to specify the PATTERN definition that is to be used for filling each node of the network. Note that if the value of the _PATTERN variable is *j* for a particular node, PROC NETDRAW uses the specifications in the *jth generated* PATTERN *definition*, not the specifications in the PATTERN*j statement*.

The patterns that can be used with PROC NETDRAW are any of the patterns that can be used for drawing bars (not ones that are used for drawing maps). However, for the text to be visible, you may want to restrict the patterns used to be empty and change only the color of the pattern. You can also use solid fills with a light color and specify the COUTLINE= and CCRITOUT= options to mark noncritical and critical nodes with different colors for the outline.

See *SAS/GRAPH Software: Reference* for details about creating, canceling, reviewing, and altering PATTERN definitions. For a brief description of the PATTERN statement and for a list of available patterns, see Chapter 8, "The GANTT Procedure."

If a PATTERN variable is not specified, the procedure uses the values of the E_FINISH and L_FINISH variables (if these variables exist in the Network data set) to determine if activities in the project are normal, critical, or supercritical. The procedure then uses the first *generated* PATTERN *definition* to fill the nodes corresponding to noncritical activities, the second *generated* PATTERN *definition* for nodes corresponding to critical activities, and the third *generated* PATTERN *definition* for nodes corresponding to supercritical activities.

For zoned network diagrams, if there is no PATTERN variable, the ZONEPAT option enables you to color the nodes based on the values of the ZONE= variable.

Using the Annotate Facility

The Annotate facility enables you to enhance graphics output produced by PROC NETDRAW. To use this facility, you must create an Annotate data set, which contains a set of graphics commands that can be superimposed on the network diagram. This data set has a specific format and must contain key variables as described in *SAS/GRAPH Software: Reference*. The chapter entitled "The Annotate Data Set" lists the variables that are required in this data set and explains the coordinate systems used by the Annotate facility. The present section explains the use of data coordinates specifically with reference to the NETDRAW procedure.

When annotating a graph produced by any of the graphics procedures, it is helpful to use data coordinates that refer to the data values corresponding to the graph that is being annotated. For example, if you want to label a particular node of a network diagram with additional text, you can position the text accurately if you use data coordinates instead of screen coordinates. With respect to PROC NETDRAW, the Annotate facility uses the _X_ and _Y_ values in the Layout data set as the basis for the data coordinate system. To use this feature, you can invoke PROC NETDRAW (with the NODISPLAY option, if necessary) for the given network to produce the Layout data set that contains the _X_ and _Y_ coordinates for each node of the network. This data set can then be used to create the required Annotate data set containing the graphics commands positioning the primitives appropriately on the diagram using the data coordinates. See Example 9.16 and Example 9.17 for illustrations of this feature.

NOTE: The data coordinate system enables you to annotate the graph even if it spans multiple pages. However, each annotation must be entirely contained within a given page. For example, you cannot annotate a line on the network diagram that runs from one page of the diagram to another.

Web-Enabled Network Diagrams

The WEB variable enables you to define a HTML reference for each activity. This HTML reference is associated with the node corresponding to the activity. The WEB variable is a character variable and the values need to be of the form "HREF=htmlpage".

In addition, you can also store the coordinate and link information defined by the WEB= option in a SAS data set by specifying the IMAGEMAP= option in the PROC NETDRAW statement. By processing this SAS data set using a DATA step, you can generate customized HTML pages for your network diagram.

Macro Variable _ORNETDR

The NETDRAW procedure defines a macro variable named _ORNETDR. This variable contains a character string that indicates the status of the procedure. It is set at procedure termination. The form of the _ORNETDR character string is STATUS= REASON= , where STATUS= is either SUCCESSFUL or ERROR_EXIT and REASON= (if PROC NETDRAW terminated unsuccessfully) can be one of the following:

CYCLE

BADDATA_ERROR

MEMORY_ERROR

IO_ERROR

SEMANTIC_ERROR

SYNTAX_ERROR

NETDRAW_BUG

UNKNOWN_ERROR

This information can be used when PROC NETDRAW is one step in a larger program that needs to determine whether the procedure terminated successfully or not. Because _ORNETDR is a standard SAS macro variable, it can be used in the ways that all macro variables can be used.

In addition to providing the "STATUS= REASON= " string that indicates the status of the procedure, the macro variable _ORNETDR also provides some information about the network diagram produced by the current invocation of PROC NETDRAW.

The information given in _ORNETDR is described in the following list, along with the keyword that identifies it. These values refer to those actually used in producing the network diagram and are not necessarily the same as those specified in the invocation of the procedure.

- HPAGES= The number of horizontal pages
- VPAGES= The number of vertical pages
- SEGNAME= The name of the first network diagram segment in graphics mode

NOTE: Some of the information might be redundant or predictable in certain display modes. For example, the value of the SEGNAME= option is empty in line-printer and full-screen modes. The values of the HPAGES= and VPAGES= options are equal to 1 in full-screen mode.

Computer Resource Requirements

There is no inherent limit on the size of the network that you can draw with the NETDRAW procedure. Naturally, a sufficient amount of core memory must be available in order to invoke and initialize the SAS system. Furthermore, the amount of memory that is required depends on the mode of invocation of the procedure. The procedure attempts to store all the data in core memory. However, if the problem is too large to fit in core memory, the procedure resorts to using utility data sets and swaps between core memory and utility data sets as necessary.

The storage requirement for the data area that the procedure requires is proportional to the number of nodes and arcs in the network. You can further increase the memory that is required by specifying the DP option in the ACTNET statement. Recall that the DP option requests the use of a dynamic programming algorithm to route the arcs between the nodes, and such algorithms tend to grow exponentially with the size of the problem being solved.

ODS Style Templates

ODS style templates, or *styles*, control the overall look of your output. An ODS style template consists of a set of *style elements*. A style element is a collection of *style attributes* that apply to a particular feature or aspect of the output. You can specify a value for each attribute in a style. See Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*), for a thorough discussion of ODS Graphics.

To create your own style or to modify a style for use with ODS Graphics, you need to understand the relationships between style elements and graph features. This information is provided in the ODS Graphics documentation at `http://support.sas.com/documentation/onlinedoc/base/`. You can create and modify style templates with the TEMPLATE procedure. For more information, see the section "TEMPLATE Procedure: Creating a Style Template" in the *SAS Output Delivery System: User's Guide*. Kuhfeld (2010) also offers detailed information and examples.

PROC NETDRAW Style Template

A predefined ODS style template named NETDRAW is available for the NETDRAW procedure. You can use the template to maintain a consistent appearance in all graphical output produced by the procedure.

To change the current style, specify the STYLE= option in an ODS destination statement. The specified style is applied to all output for that destination until you change or close the destination or start a new SAS session. For example, the following statement specifies that ODS should apply the NETDRAW style template to all HTML output:

```
ods html style=netdraw;
```

To disable the use of graphical styles, specify the SAS system option NOGSTYLE.

The parent style template for the NETDRAW style is the DEFAULT style. Table 9.7 lists the style elements (in bold) and corresponding attributes specified in the NETDRAW style. The table also indicates which, if any, PROC NETDRAW options or graphics options (in a GOPTIONS statement) can be used to override the value of a style attribute.

Table 9.7 Style Elements and Attributes in the NETDRAW Style

Element/Attributes	Description	NETDRAW Option	GOPTION
GraphColors	Colors of various graph features		
gdata1	Noncritical nodes or nodes in the first zone	PATTERN=, ZONEPAT	CPATTERN=, COLORS=
gdata2	Critical nodes or nodes in the second zone	PATTERN=, ZONEPAT	CPATTERN=, COLORS=
gdata3	Nodes in the third zone	PATTERN=, ZONEPAT	CPATTERN=, COLORS=
gdata4	Nodes in the fourth zone	PATTERN=, ZONEPAT	CPATTERN=, COLORS=
gdata5	Nodes in the fifth zone	PATTERN=, ZONEPAT	CPATTERN=, COLORS=
gdata6	Nodes in the sixth zone	PATTERN=, ZONEPAT	CPATTERN=, COLORS=
gdata7	Nodes in the seventh zone	PATTERN=, ZONEPAT	CPATTERN=, COLORS=
gdata8	Nodes in the eighth zone	PATTERN=, ZONEPAT	CPATTERN=, COLORS=
gdata9	Nodes in the ninth zone	PATTERN=, ZONEPAT	CPATTERN=, COLORS=
gdata10	Nodes in the tenth zone	PATTERN=, ZONEPAT	CPATTERN=, COLORS=
gdata11	Nodes in the eleventh zone	PATTERN=, ZONEPAT	CPATTERN=, COLORS=
gdata12	Nodes in the twelfth zone	PATTERN=, ZONEPAT	CPATTERN=, COLORS=
gaxis	Borderlines		COLORS=
greferencelines	Horizontal and vertical reference lines		COLORS=
gtext	Text		CTEXT=
gtextt	Title		CTITLE=
gcdata	Arcs		COLORS=
GraphFonts	Fonts for various graph features		
GraphDataFont	Default		FTEXT=
GraphLabelFont	Annotation text		FTEXT=
GraphTitleFont	Title text		FTITLE=
GraphAxisLines	Attributes related to graph axes		
Color	GraphColors('gaxis')	CAXIS=	COLORS=
GraphConnectLine	Attributes related to arcs		
Color	GraphColors('gcdata')	CARCS=, CCRITARCS=	COLORS=
GraphReference	Attributes related to horizontal and vertical reference lines		
Color	GraphColors('greferencelines')	CREF=, CREFBRK=	COLORS=
GraphDataText	Attributes related to general text		
Color	GraphColors('gtext')	CTEXT=	CTEXT=
Font	GraphFonts('GraphDataFont')	FONT=	FTEXT=

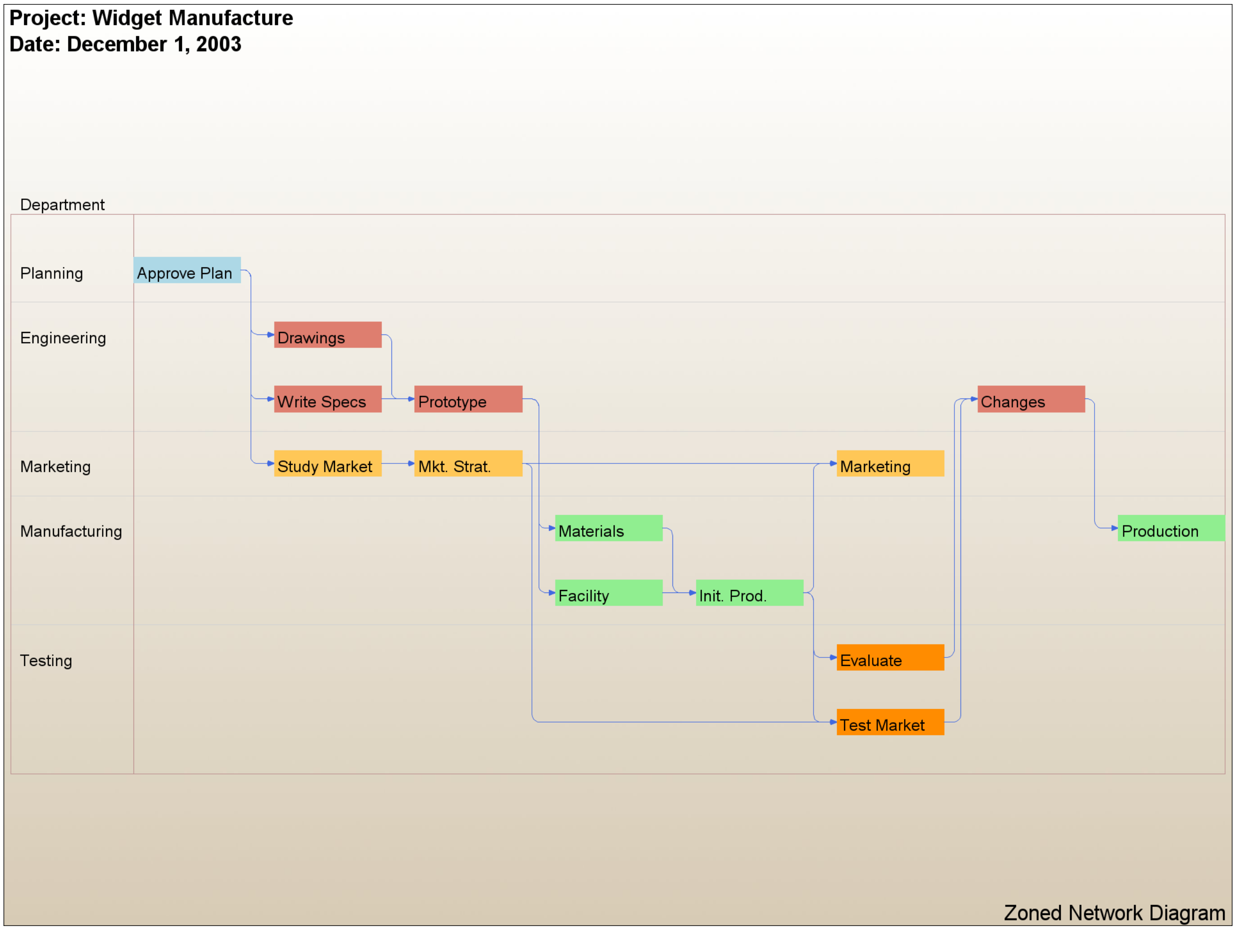
Table 9.7 (continued)

Element/Attributes	Description	NETDRAW Option	GOPTION
GraphTitleText	Attributes related to title text		
Color	GraphColors('gtextt')		CTITLE=
Font	GraphFonts('GraphTitleFont')		FTITLE=
GraphTitle1Text	Attributes related to the first title text		
Color	GraphColors('gtextt')		CTITLE=
Font	GraphFonts('GraphTitleFont')		FTITLE=
GraphLabelText	Attributes related to annotation text		
Color	GraphColors('glabel')		CTEXT=
Font	GraphFonts('GraphLabelFont')		FTEXT=
GraphDataDefault	Default values for the attributes specified in Table 9.8		
Color	GraphColors('gdata')		COLORS=
GraphBackground	Attributes related to graph background		
Image	Background image		CBACK=

Attributes that you do not override retain the values specified in the style template.

Figure 9.9 demonstrates features of the NETDRAW graphical style. The diagram in the figure is the first output from Example 9.11.

Figure 9.9 NETDRAW Style Template: Example



Default Values

If the SAS system option GSTYLE is in effect (this is the default), then the default values of certain PROC NETDRAW options can depend on the current ODS style template. Table 9.8 lists these PROC NETDRAW options and lists the order in which PROC NETDRAW searches for each option's default value. The order assumes that the GSTYLE system option is in effect; if that is not the case, then the steps that refer to ODS style templates are ignored. Names with arguments indicate style elements and attributes of the current ODS style template. For example, "GraphAxisLines('Color')" refers to the Color attribute of the GraphAxisLines element.

Table 9.8 PROC NETDRAW Options: Search Orders for Default Values

Option	Search Order for Default Value
CARCS=	1. GraphConnectLine(Color) 2. The fourth color in the COLORS= list in the GOPTIONS statement
CAXIS=	1. GraphAxisLines(Color) 2. GraphDataDefault(Color) 3. The first color in the COLORS= list in the GOPTIONS statement
CREF=	1. GraphReference(Color) 2. GraphDataDefault(Color) 3. The first color in the COLORS= list in the GOPTIONS statement
CREFBRK=	1. GraphReference(Color) 2. GraphDataDefault(Color) 3. The first color in the COLORS= list in the GOPTIONS statement
CTEXT=	1. The value specified for the CTEXT= option in the GOPTIONS statement 2. GraphDataText(Color) 3. GraphDataDefault(Color) 4. The first color in the COLORS= list in the GOPTIONS statement
FONT=	1. The value specified for the FTEXT= option in the GOPTIONS statement 2. GraphDataText(Font) 3. The default hardware font for the graphics output device

Examples: NETDRAW Procedure

This section contains 18 examples that illustrate several features of the NETDRAW procedure. Most of the examples use the data from the Widget Manufacturing Project described in Chapter 4, "The CPM Procedure." Two tables, Table 9.9 and Table 9.10, at the end of this section list all the examples in this chapter and the options and statements in the NETDRAW procedure that are illustrated by each example.

Example 9.1: Line-Printer Network Diagram

This example uses the data set WIDGET that was used in Example 4.2 in Chapter 4, "The CPM Procedure," to illustrate the Activity-on-Node representation of the project. The following program invokes PROC NETDRAW twice. First, the activity data set WIDGET is used as input to the procedure. The activity and successor information is identified using the ACTIVITY= and SUCCESSOR= options in the ACTNET statement. The LINEPRINTER option is specified, producing the line-printer network diagram shown in Output 9.1.1.

```
data widget;
   format task $12. succ1-succ3 $12.;
   input task & days succ1 & succ2 & succ3 & ;
   datalines;
Approve Plan    5  Drawings      Study Market  Write Specs
Drawings       10  Prototype     .             .
Study Market    5  Mkt. Strat.   .             .
Write Specs     5  Prototype     .             .
Prototype      15  Materials     Facility      .
Mkt. Strat.    10  Test Market   Marketing     .
Materials      10  Init. Prod.   .             .
Facility       10  Init. Prod.   .             .
Init. Prod.    10  Test Market   Marketing     Evaluate
Evaluate       10  Changes       .             .
Test Market    15  Changes       .             .
Changes         5  Production    .             .
Production      0  .             .             .
Marketing       0  .             .             .
;

title 'Widget Manufacture';
options ps=32 ls=78;
proc netdraw data=widget lineprinter;
   actnet / activity=task successor=(succ1 succ2 succ3);
   run;
```

Output 9.1.1 Line-Printer Network Diagram

Widget Manufacture

```
                   --------------                        --------------
                -->|Drawings    |---                  -->|Materials   |---
                |  --------------  |                  |  --------------  |
                |                  |                  |                  |
                |                  |                  |                  |
                |                  |                  |------------------+--
                |                  |                  |                  |
                |                  |                  |                  |
--------------  |  --------------  |  --------------  |  --------------  |
|Approve Plan|--+->|Write Specs |---->|Prototype   |--+->|Facility    |---->
--------------  |  --------------     --------------  |  --------------
                |                                     |
                |                                     |
                |                                     |
                |                                     |
                |                                     |
                |  --------------     --------------  |
                -->|Study Market|---->|Mkt. Strat. |------------------------
                   --------------     --------------
```

Output 9.1.1 *continued*

Widget Manufacture

```
                 -------------
              -->|Test Market |---
              |  -------------   |
              |                  |
              |                  |
----------------|                |
              |                  |
              |                  |
-------------  |  -------------  |  -------------     -------------
|Init. Prod. |--+->|Evaluate    |---->|Changes     |---->|Production  |
-------------  |  -------------     -------------     -------------
              |
              |
              |
              |
              |
              |  -------------
----------------->|Marketing   |
                 -------------
```

Next, PROC CPM is invoked to schedule the project, and the resulting Schedule data set is used as input to the NETDRAW procedure. In addition to the ACTIVITY= and SUCCESSOR= options, the DURATION= option is used in the ACTNET statement. The DURATION= option adds the values of the DURATION variable within each node of the network. The procedure also displays the values of the E_START, E_FINISH, L_START, L_FINISH, T_FLOAT, and F_FLOAT variables within each node. The network is displayed in Output 9.1.2.

```
proc cpm data=widget out=sched
         date='1dec03'd;
   activity task;
   successor succ1 succ2 succ3;
   duration days;
   run;

options ps=45 ls=90;
title2 'Schedule Information';
proc netdraw data=sched lineprinter;
   actnet / activity=task
            successor=(succ1 succ2 succ3)
            duration = days;
   run;
```

Output 9.1.2 Project Schedule

Widget Manufacture
Schedule Information

```
                              -------------------------
                              |Drawings          Dur:10|
                           -->|ES:06DEC03   EF:15DEC03|---                             -->
                           |  |LS:06DEC03   LF:15DEC03|  |                             |
                           |  |TF: 0         FF: 0     |  |                             |
                           |  -------------------------  |                             |
                           |                             |                             |
                           |                             |                             |
                           |                             |                             |--
                           |                             |                             |
                           |                             |                             |
-------------------------  |  -------------------------  |  -------------------------  |
|Approve Plan      Dur: 5|  |  |Write Specs       Dur: 5|  |  |Prototype         Dur:15|  |
|ES:01DEC03   EF:05DEC03|--+->|ES:06DEC03   EF:10DEC03|---->|ES:16DEC03   EF:30DEC03|--+->
|LS:01DEC03   LF:05DEC03|  |  |LS:11DEC03   LF:15DEC03|     |LS:16DEC03   LF:30DEC03|  |
|TF: 0         FF: 0     |  |  |TF: 5         FF: 5     |     |TF: 0         FF: 0     |  |
-------------------------  |  -------------------------     -------------------------  |
                           |                                                           |
                           |                                                           |
                           |                                                           |
                           |                                                           |
                           |                                                           |
                           |  -------------------------     -------------------------  |
                           |  |Study Market      Dur: 5|     |Mkt. Strat.       Dur:10|  |
                           -->|ES:06DEC03   EF:10DEC03|---->|ES:11DEC03   EF:20DEC03|-----
                              |LS:05JAN04   LF:09JAN04|     |LS:10JAN04   LF:19JAN04|
                              |TF:30         FF: 0     |     |TF:30         FF:30     |
                              -------------------------     -------------------------
```

Output 9.1.2 *continued*

Widget Manufacture
Schedule Information

```
-------------------------                                   -------------------------
|Materials        Dur:10|                                   |Test Market      Dur:15|
|ES:31DEC03   EF:09JAN04|---                             -->|ES:20JAN04   EF:03FEB04|---
|LS:31DEC03   LF:09JAN04|  |                             |  |LS:20JAN04   LF:03FEB04|  |
|TF: 0        FF: 0     |  |                             |  |TF: 0        FF: 0     |  |
-------------------------  |                             |  -------------------------  |
                           |                             |                             |
                           |                             |                             |
---------------------------+-----------------------------|                             |
                           |                             |                             |
                           |                             |                             |
-------------------------  |  -------------------------  |  -------------------------  |
|Facility         Dur:10|  |  |Init. Prod.      Dur:10|  |  |Evaluate         Dur:10|  |
|ES:31DEC03   EF:09JAN04|---->|ES:10JAN04   EF:19JAN04|--+->|ES:20JAN04   EF:29JAN04|---->
|LS:31DEC03   LF:09JAN04|     |LS:10JAN04   LF:19JAN04|  |  |LS:25JAN04   LF:03FEB04|
|TF: 0        FF: 0     |     |TF: 0        FF: 0     |  |  |TF: 5        FF: 5     |
-------------------------     -------------------------  |  -------------------------
                                                         |
                                                         |
                                                         |
                                                         |
                                                         |
                                                         |  -------------------------
                                                         |  |Marketing        Dur: 0|
----------------------------------------------------------->|ES:20JAN04   EF:20JAN04|
                                                            |LS:09FEB04   LF:09FEB04|
                                                            |TF:20        FF:20     |
                                                            -------------------------
```

Output 9.1.2 *continued*

Widget Manufacture
Schedule Information

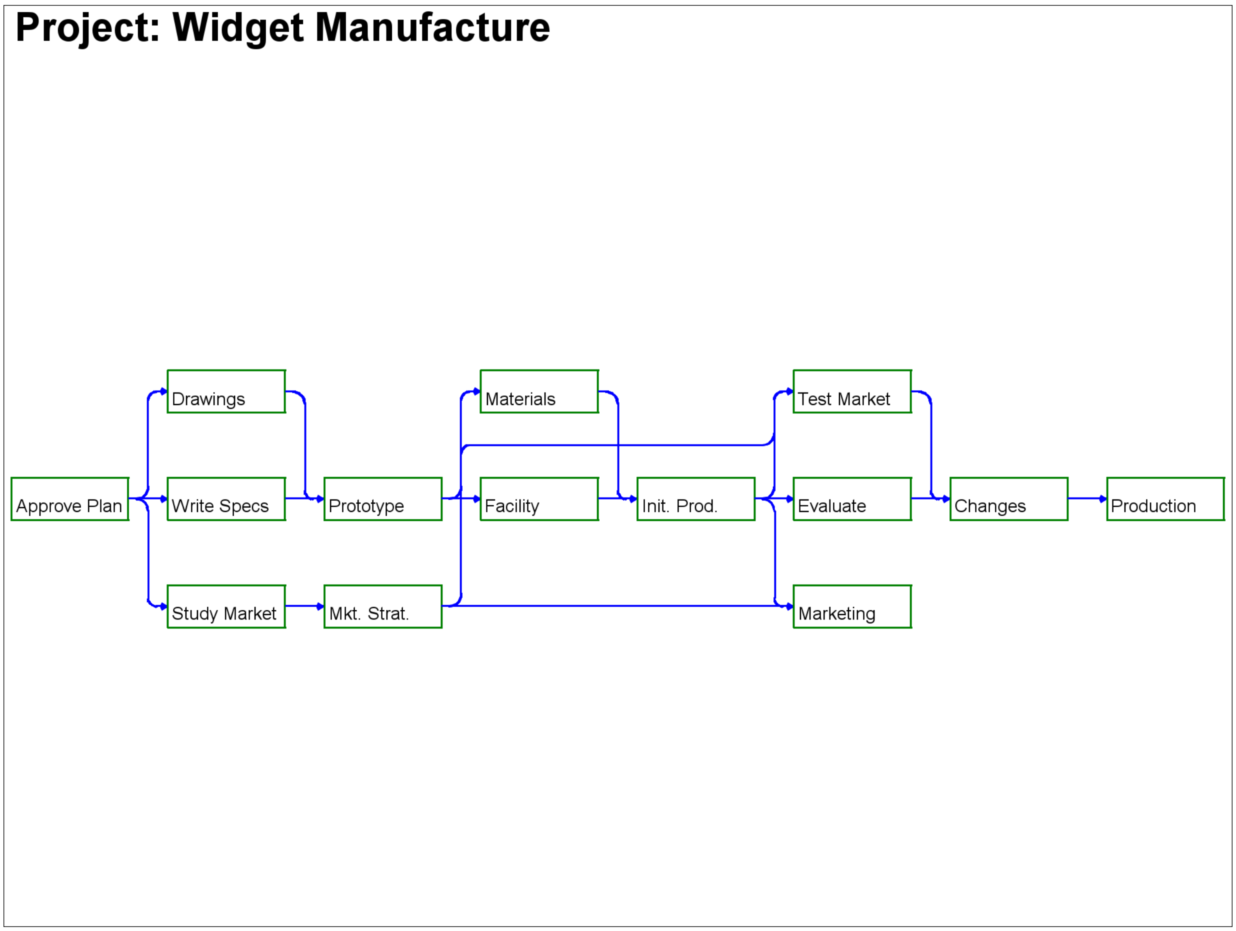
```
-------------------------     -------------------------
|Changes          Dur: 5|     |Production       Dur: 0|
|ES:04FEB04   EF:08FEB04|---->|ES:09FEB04   EF:09FEB04|
|LS:04FEB04   LF:08FEB04|     |LS:09FEB04   LF:09FEB04|
|TF: 0        FF: 0     |     |TF: 0        FF: 0     |
-------------------------     -------------------------
```

Example 9.2: Graphics Version of PROC NETDRAW

The same network used in Example 9.1 is drawn here with the GRAPHICS option (which is also the default mode of output for the NETDRAW procedure). In this example, the network is drawn before scheduling the project with PROC CPM. The global options HPOS= and VPOS= are set to 100 and 70, respectively. These options control the number of character cell positions on the screen. The network is displayed in Output 9.2.1. Note that all the nodes are the same color as specified by the PATTERN statement, the color of the arcs is blue as specified by the CARCS= option, the width of all lines is 3 as specified by the LWIDTH= option.

```
goptions hpos=100 vpos=70 border;
pattern1 c=green v=e;
title h=3  j=l ' Project: Widget Manufacture';
proc netdraw data=widget graphics;
   actnet / act=task succ=(succ1 succ2 succ3)
            height=1.5 compress
            carcs=blue lwidth=3;
   run;
```

Output 9.2.1 Project Network



Example 9.3: Spanning Multiple Pages

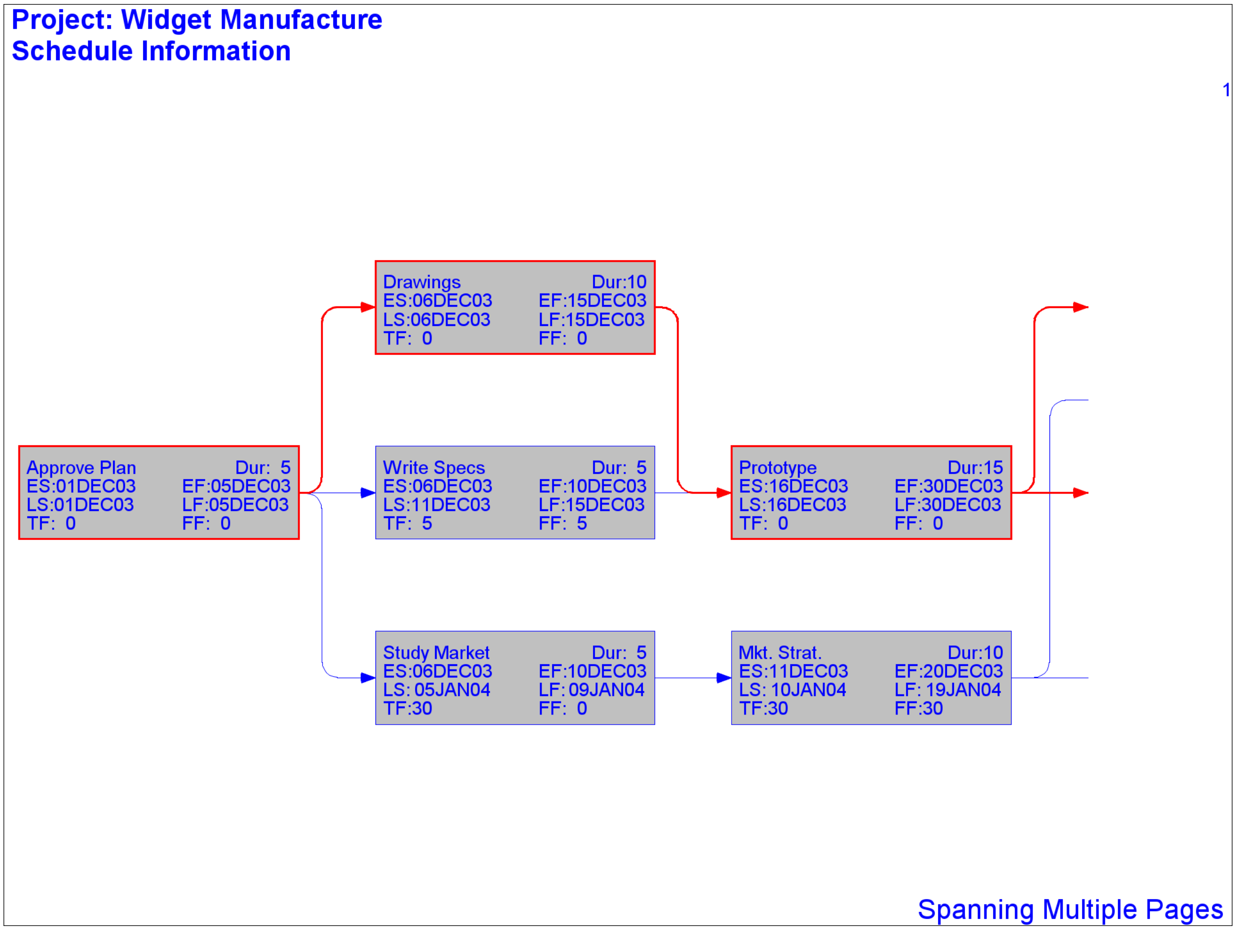
In this example, the Schedule data set produced by PROC CPM is displayed in a graphics network. As in the second part of Example 9.1, the procedure displays the duration as well as the early and late start and finish times and the total float and free float of each activity in the node corresponding to that activity. The network cannot fit on one page and is drawn across three pages, as shown in Output 9.3.1.

This example also illustrates several options for controlling the appearance of the network diagram. The pattern statements set a background fill color for all the nodes of the network (PATTERN1 and PATTERN2). The COUTLINE= and CCRITOUT= options set the outline colors for noncritical and critical activities, respectively. Recall that the procedure uses the values of the E_FINISH and L_FINISH variables to determine if an activity is critical. The CARCS= and CCRITARCS= options color the regular arcs blue and the critical arcs (arcs connecting critical activities) red, respectively and the CTEXT= options sets the color of the text to blue. Finally, the LWIDTH= option sets the default width for all the lines in the network, and the LWCRIT= option further highlights the critical arcs by drawing them with thicker lines.

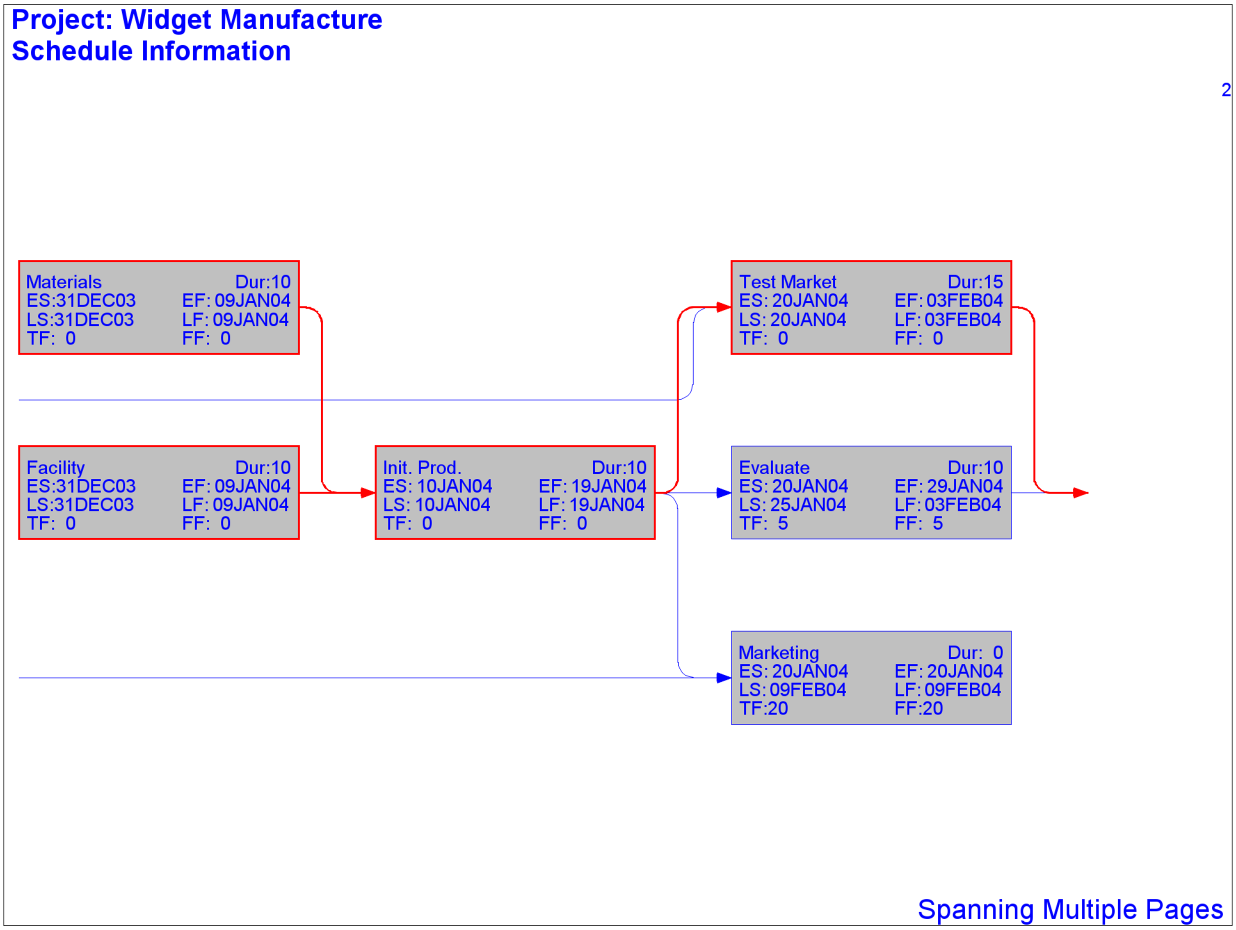
In this invocation of PROC NETDRAW, the SEPARATEARCS option is used so that the two parallel arcs leading into the activity 'Test Market' (one from 'Mkt.Strat.' and the other from 'Init. Prod.') are drawn along separate tracks instead of along a single track as in Example 9.2.

```
 /* Activity-on-Node representation of the project */
data widget;
   format task $12. succ1-succ3 $12.;
   input task & days succ1 & succ2 & succ3 & ;
   datalines;
Approve Plan   5  Drawings      Study Market  Write Specs
Drawings      10  Prototype     .             .
Study Market   5  Mkt. Strat.   .             .
Write Specs    5  Prototype     .             .
Prototype     15  Materials     Facility      .
Mkt. Strat.   10  Test Market   Marketing     .
Materials     10  Init. Prod.   .             .
Facility      10  Init. Prod.   .             .
Init. Prod.   10  Test Market   Marketing     Evaluate
Evaluate      10  Changes       .             .
Test Market   15  Changes       .             .
Changes        5  Production    .             .
Production     0  .             .             .
Marketing      0  .             .             .
;

goptions hpos=80 vpos=50 border;
pattern1 c=ltgray v=s;
pattern2 c=ltgray v=s;

title c=blue j=l h=1.5 ' Project: Widget Manufacture';
title2 c=blue j=l h=1.5 ' Schedule Information';
footnote c=blue j=r h=1.5 'Spanning Multiple Pages ';
proc netdraw data=sched graphics;
   actnet / act=task
            succ=(succ1 succ2 succ3)
            dur = days
            coutline=blue
            ccritout=red
            carcs=blue
            ccritarcs=red
            ctext=blue
            lwidth=1
            lwcrit=2
            separatearcs;
   run;
```

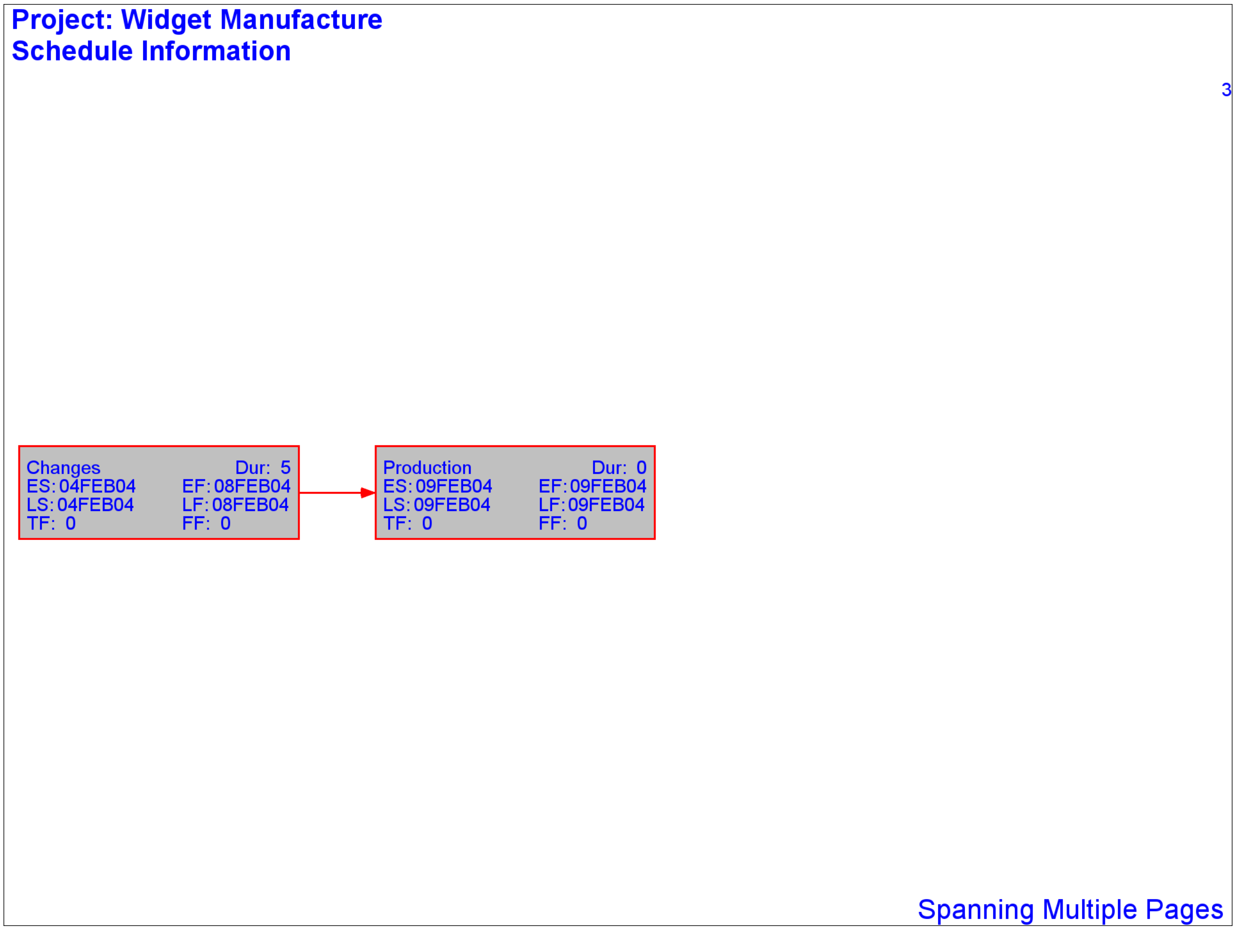
Output 9.3.1 Project Schedule



Output 9.3.1 *continued*



Output 9.3.1 *continued*

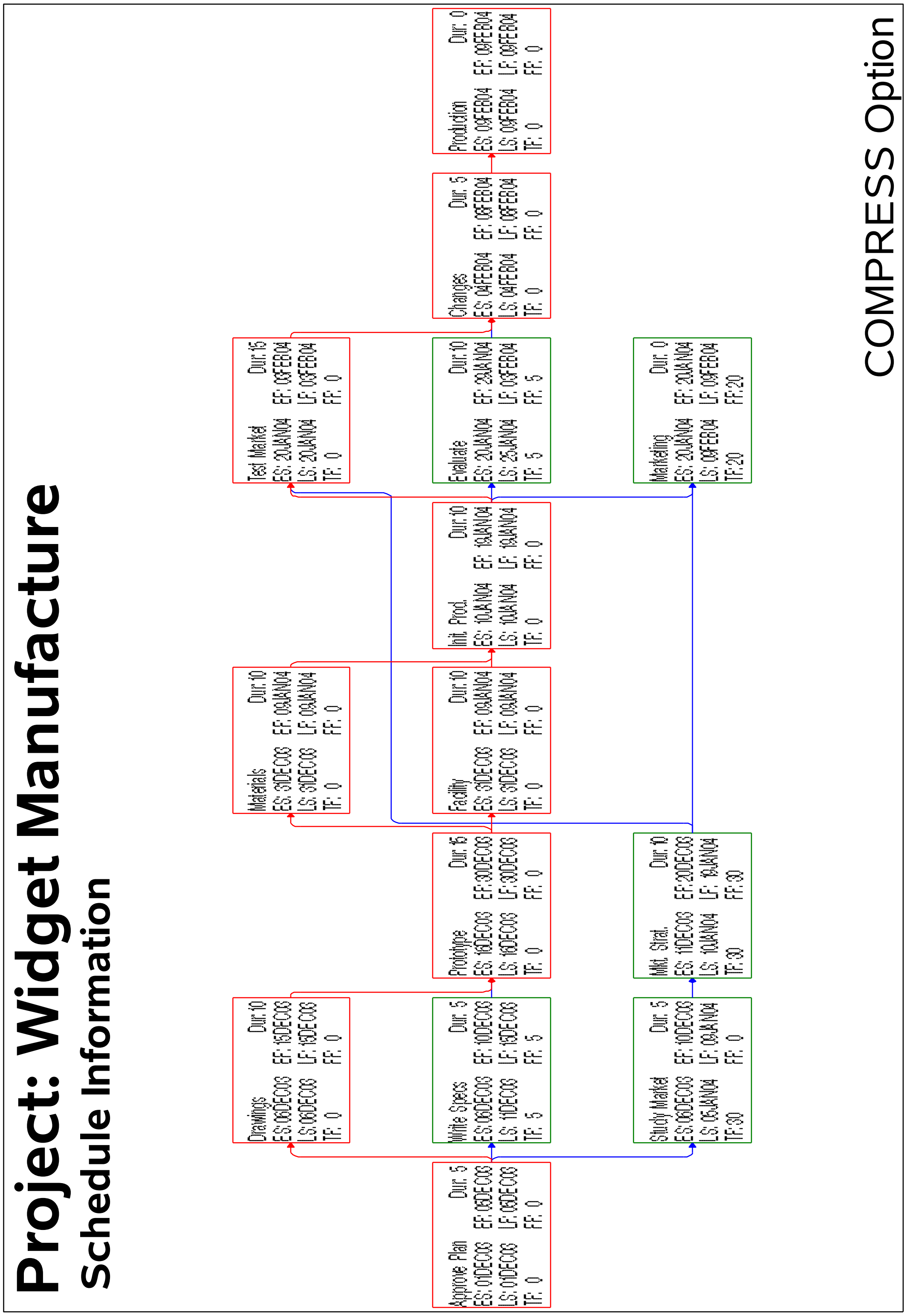


Example 9.4: The COMPRESS and PCOMPRESS Options

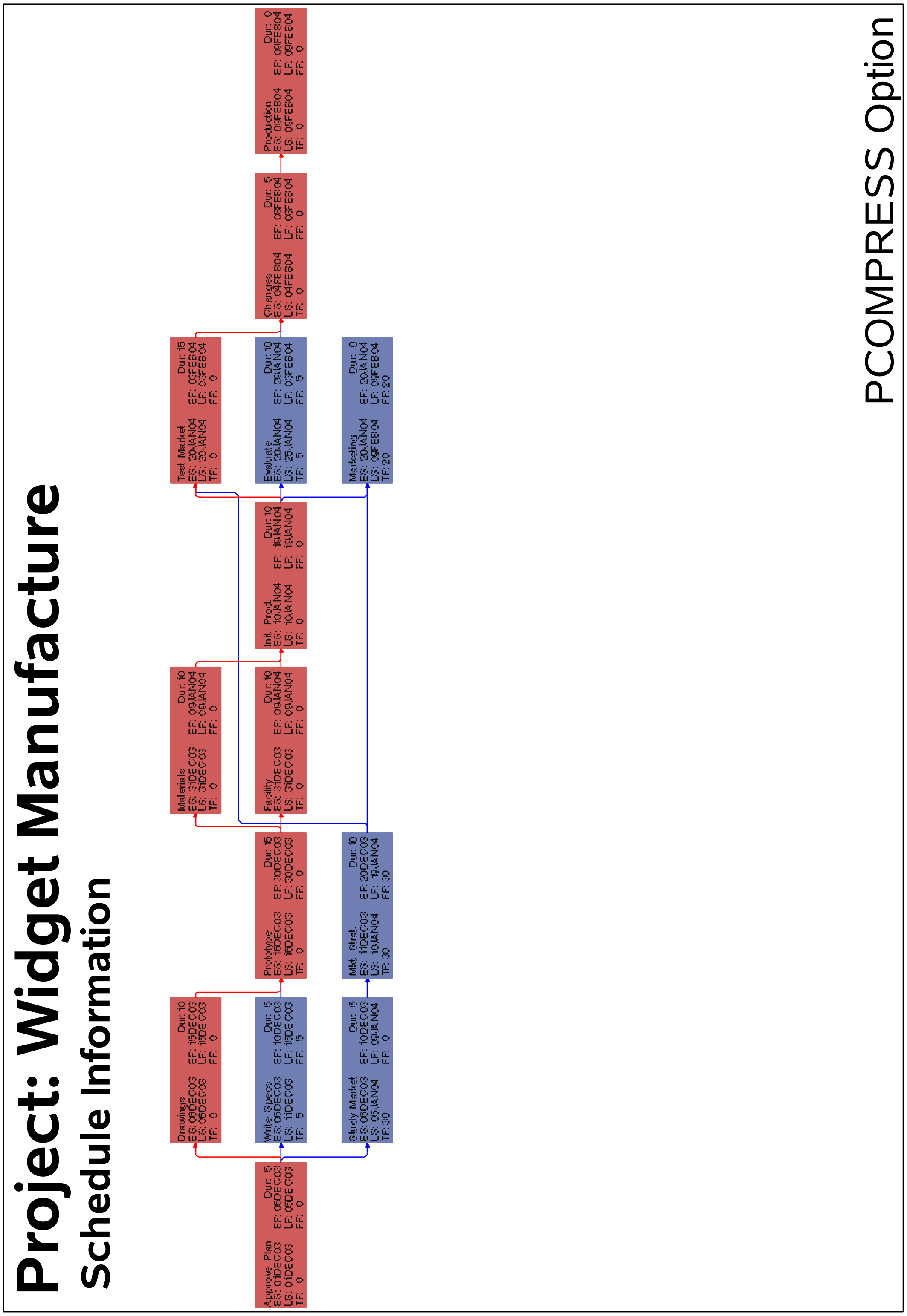
In Example 9.2, the number of character cell positions were specified so that the entire network fit on one page; in Example 9.3, the network spanned several pages. To force the network diagram to fit on one page automatically, you can use the COMPRESS or PCOMPRESS options. Both options use window transformations to fit the network on one page: the COMPRESS option treats the horizontal and vertical transformations independently of each other so that one direction may not be as compressed as the other; the PCOMPRESS option uses a proportional transformation so that each direction is compressed as much as the other. The following two invocations of PROC NETDRAW illustrate the differences in the diagrams produced.

In both network diagrams, PATTERN statements are used to color the nodes red or green, depending on whether the activities they represent are critical or not. The first PATTERN statement is for nodes corresponding to noncritical activities, and the second statement is for critical activities. The second invocation of PROC NETDRAW also uses the NOVCENTER option in the ACTNET statement to turn off centering in the vertical direction, so that the network is drawn immediately below the titles.

Output 9.4.1 Project Network: COMPRESS Option



Output 9.4.2 Project Network: PCOMPRESS Option



```
goptions border;

pattern1 v=e c=green;
pattern2 v=e c=red;

title j=l h=3 ' Project: Widget Manufacture';
title2 j=l h=2 '  Schedule Information';
footnote j=r h=2 'COMPRESS Option ';
proc netdraw data=sched graphics;
   actnet / act=task
            succ=(succ1 succ2 succ3)
            dur = days
            carcs=blue
            ccritarcs=red
            separatearcs
            font=swiss
            height=1.5
            compress;
   run;

footnote j=r h=2 'PCOMPRESS Option ';
proc netdraw data=sched graphics;
   actnet / act=task
            succ=(succ1 succ2 succ3)
            dur = days
            carcs=blue
            ccritarcs=red
            separatearcs
            font=swiss
            height=1.5
            pcompress
            novcenter;
   run;
```

Example 9.5: Controlling the Display Format

In addition to the COMPRESS and PCOMPRESS options and the HPOS= and VPOS= global options, you can control the amount of information displayed on a single page by

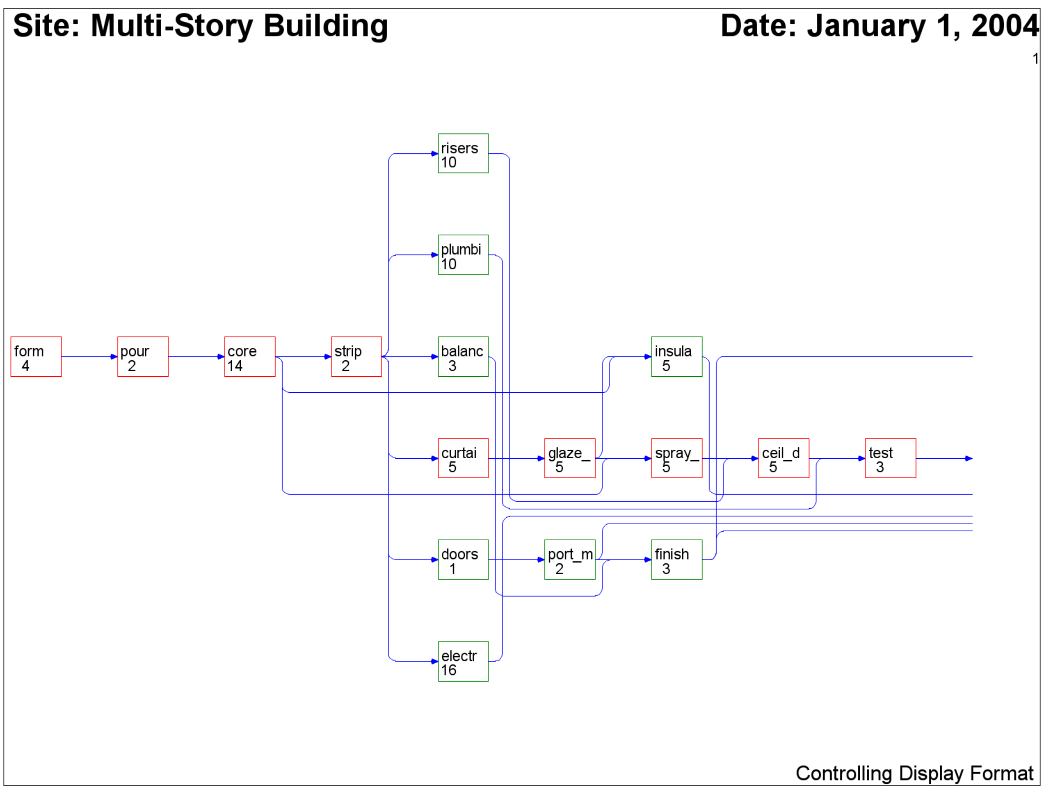
- turning off the default ID variable selection (using the NODEFID option) and using the ID= option to select only a few variables of interest in the Network data set
- setting the dimensions of each node using the BOXWIDTH= and the BOXHT= options
- specifying the horizontal and vertical distances between nodes using the XBETWEEN= and YBETWEEN= options, respectively

This example uses the data from Example 8.1 in Chapter 8, "The GANTT Procedure," and some of the options just mentioned to produce the network diagram shown in Output 9.5.1. The ID= and NODEFID options are used to display only the activity name and duration values within each node. The NOLABEL option suppresses the display of the variable names within each node. Some of the activity names are truncated by the BOXWIDTH= option. Even with the restrictions imposed, the network is too large to fit on one page and spans two pages. Note that on devices with higher resolution, you can increase the values of HPOS and VPOS (still maintaining readability) to enable more of the network to be drawn on one page.

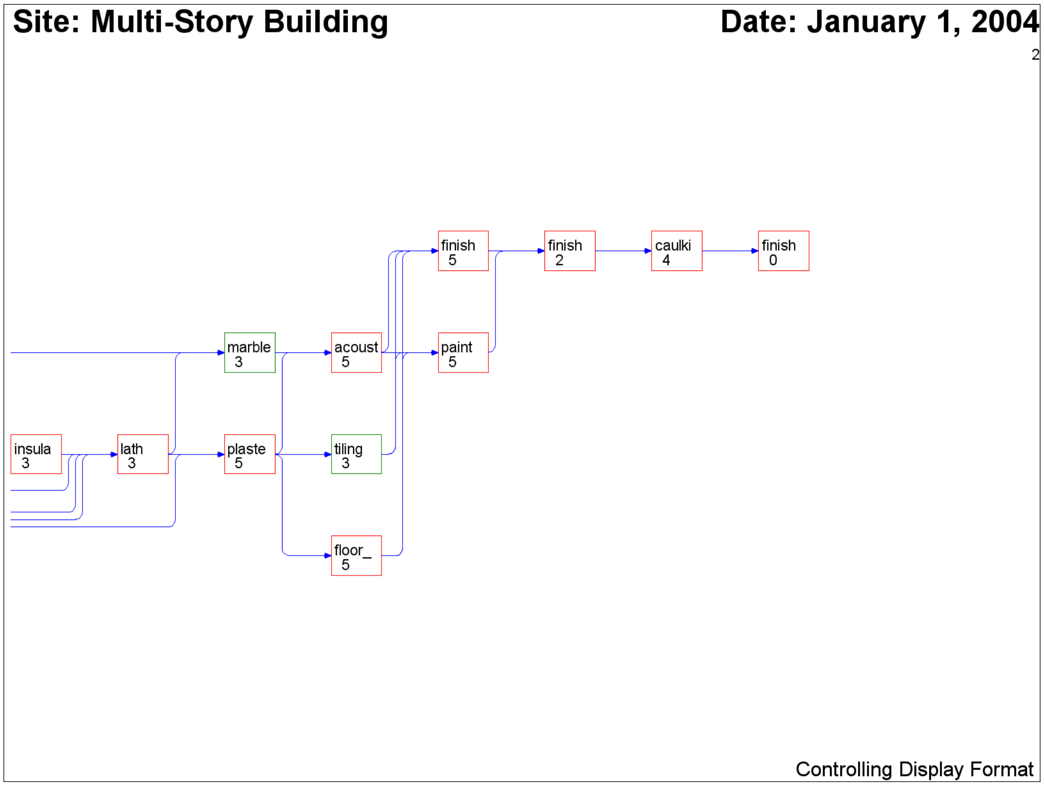
```
data ex;
   format activity $20. success1 $20. success2 $20. success3 $20.
                        success4 $20.;
   input activity dur success1-success4;
   datalines;
form                  4 pour . . .
pour                  2 core . . .
core                 14 strip spray_fireproof insulate_walls .
strip                 2 plumbing curtain_wall risers doors
strip                 2 electrical_walls balance_elevator . .
curtain_wall          5 glaze_sash . . .
glaze_sash            5 spray_fireproof insulate_walls . .
spray_fireproof       5 ceil_ducts_fixture . . .
ceil_ducts_fixture    5 test . . .
plumbing             10 test . . .
test                  3 insulate_mechanical . . .
insulate_mechanical  3 lath . . .
insulate_walls        5 lath . . .
risers               10 ceil_ducts_fixture . . .
doors                 1 port_masonry . . .
port_masonry          2 lath finish_masonry . .
electrical_walls     16 lath . . .
balance_elevator      3 finish_masonry . . .
finish_masonry        3 plaster marble_work . .
lath                  3 plaster marble_work . .
plaster               5 floor_finish tiling acoustic_tiles .
marble_work           3 acoustic_tiles . . .
acoustic_tiles        5 paint finish_mechanical . .
tiling                3 paint finish_mechanical . .
floor_finish          5 paint finish_mechanical . .
paint                 5 finish_paint . . .
finish_mechanical     5 finish_paint . . .
finish_paint          2 caulking_cleanup . . .
caulking_cleanup      4 finished . . .
finished              0 . . . .
;


proc cpm finishbefore date='1jan04'd out=sched;
   activity activity;
   duration dur;
   successors success1-success4;
   run;

proc sort;
   by e_start;
   run;
```

```
pattern1 v=e c=green;
pattern2 v=e c=red;

title j=l h=3  ' Site: Multi-Story Building'
      j=r ' Date: January 1, 2004';
footnote j=r h=2 'Controlling Display Format ';

proc netdraw data=sched graphics;
   actnet / act = activity
            succ = (success1-success4)
            id = ( activity dur )
            nolabel nodefaultid
            boxwidth = 6
            ybetween = 6
            separatearcs;
run;
```

Output 9.5.1 Controlling the Display Format



Output 9.5.1 *continued*



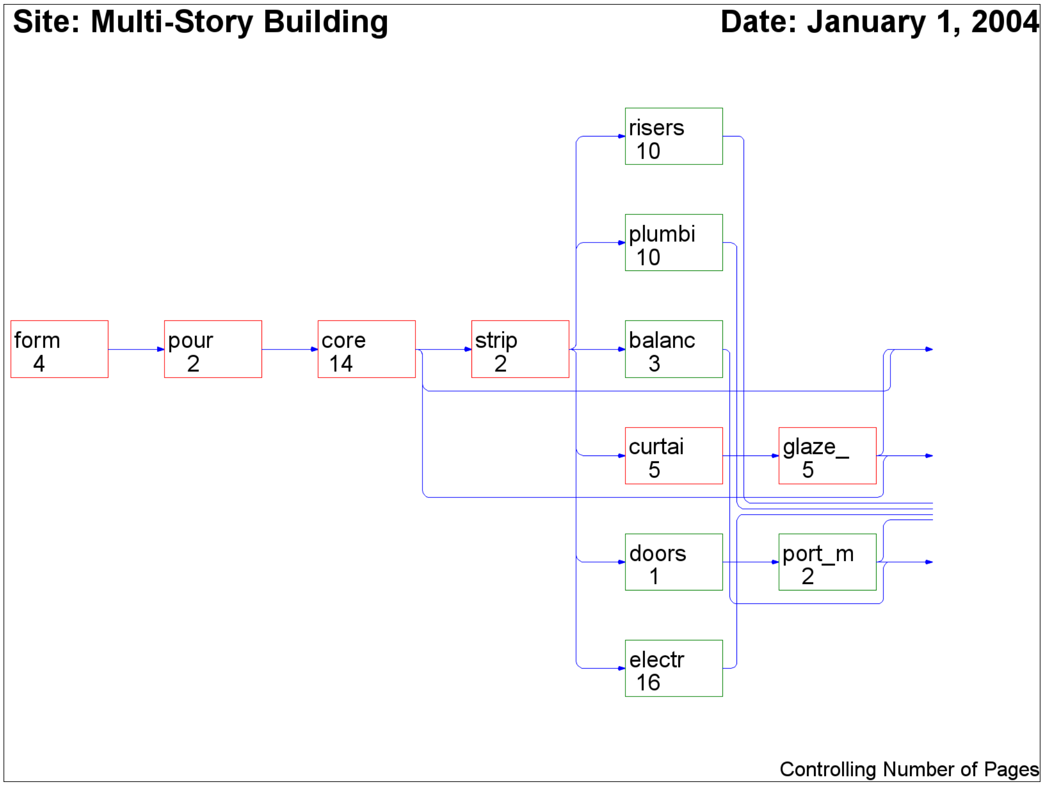
You can also control the format of the display by specifying the number of pages into which the network diagram should be split. You can do this by

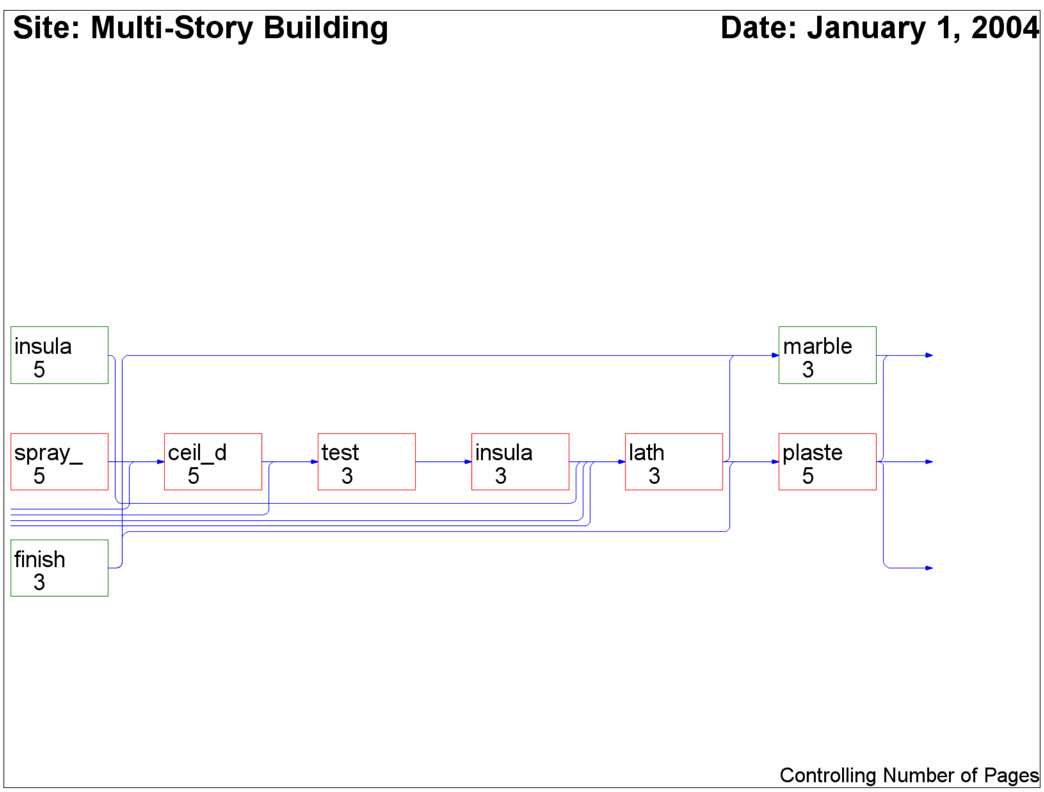
- using the HPAGES= and VPAGES= options, which specify the number of pages in the horizontal and vertical directions, respectively
- setting the number of nodes in the horizontal and vertical directions using the NXNODES= and NYNODES= options, respectively

The following statements invoke PROC NETDRAW with some additional page control options. The HTEXT= option is also used to control the height of the text used in the diagram.

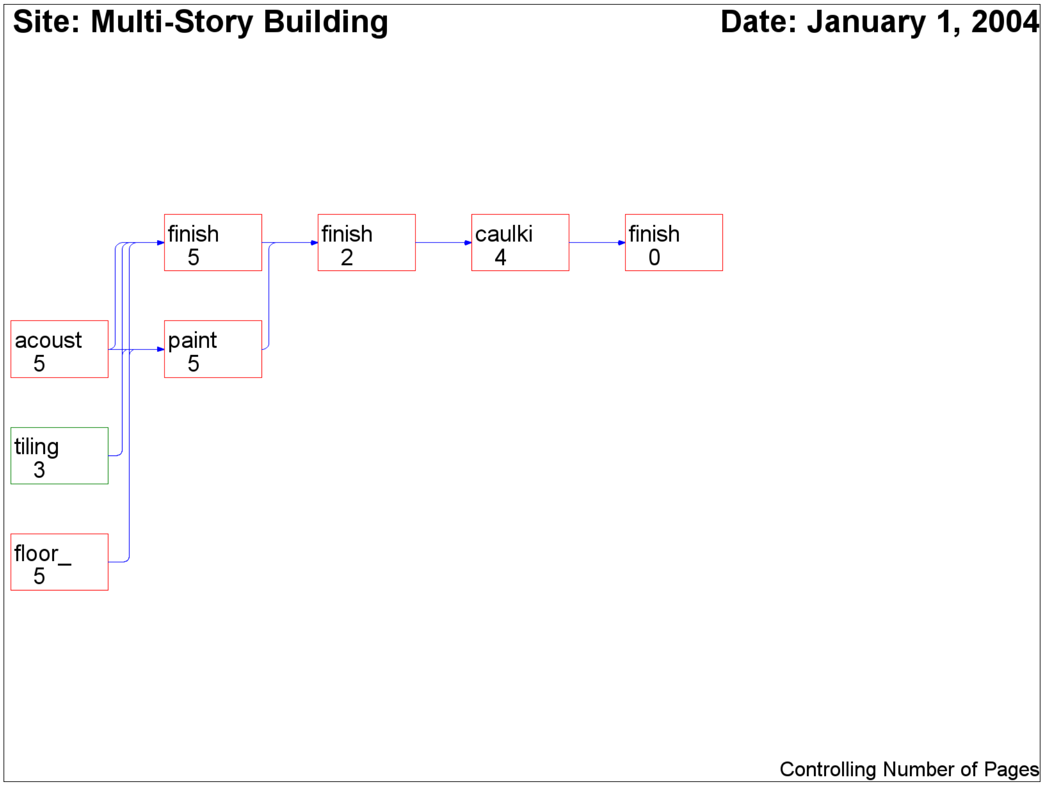
```
footnote j=r h=2 'Controlling Number of Pages';

proc netdraw data=sched graphics;
   actnet / act = activity
            succ = (success1-success4)
            id = ( activity dur )
            nolabel nodefaultid
            boxwidth = 6
            ybetween = 6
            separatearcs
            htext=2
            nopagenumber
            hpages=3 vpages=1;
run;
```

Output 9.5.2 Controlling the Number of Pages





Output 9.5.2 *continued*



Example 9.6: Nonstandard Precedence Relationships

This example illustrates the use of the LAG= option to indicate nonstandard precedence relationships between activities. Consider the widget manufacturing project described in the earlier examples. Some of the precedence constraints between the activities may be nonstandard or have a nonzero lag value. For example, the activity 'Init. Prod.' may not be able to start until 2 days after the completion of the activity 'Facility.' The Network data set is displayed in Output 9.6.1. The variable lagdur indicates the type of relationship between the activities specified in the variables task and succ.

The following statements invoke PROC NETDRAW with the LAG= option. The resulting network diagram is shown in Output 9.6.2.

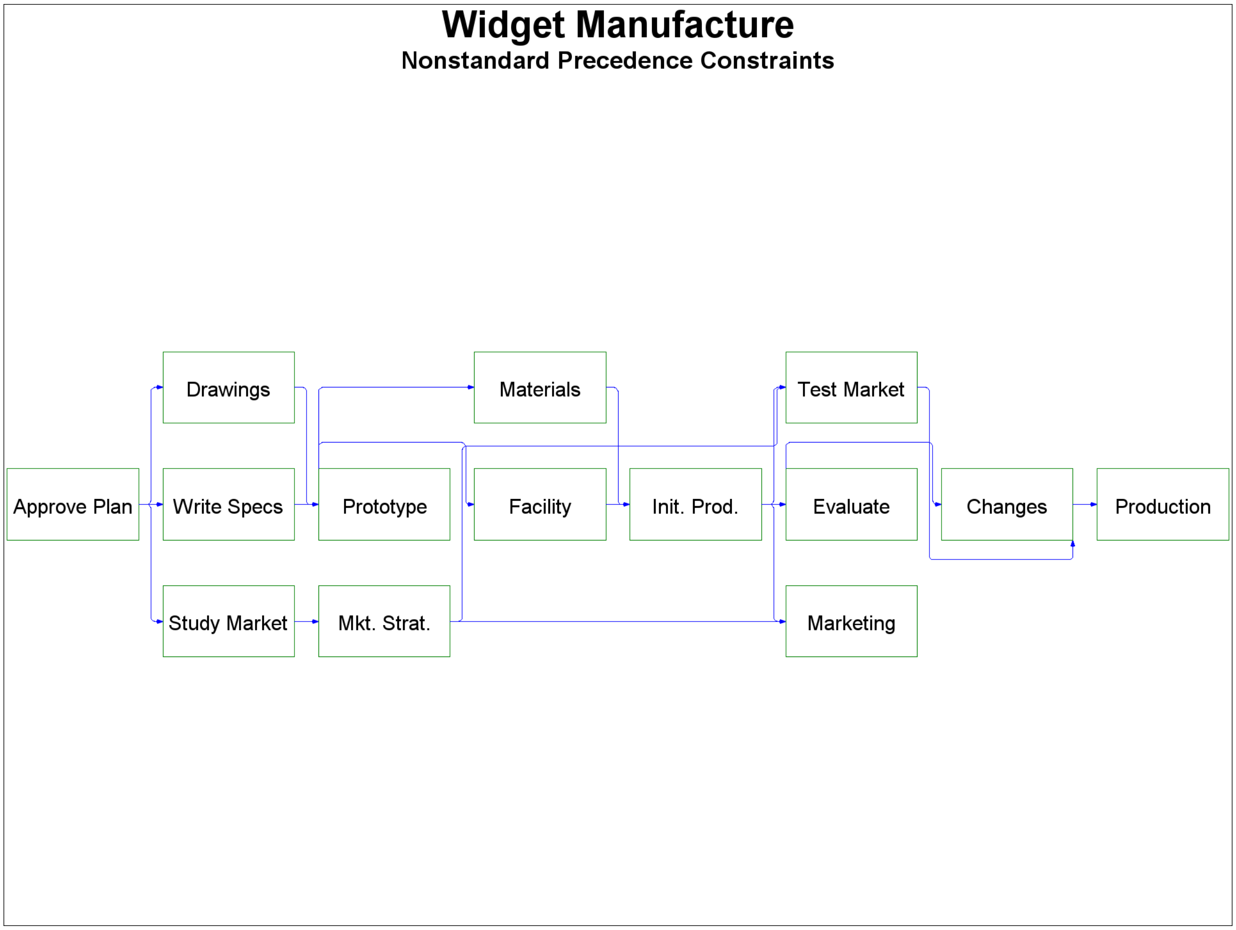
```
pattern1 v=e c=green;

title h=3 'Widget Manufacture';
title2 h=2 'Nonstandard Precedence Constraints';

proc netdraw graphics data=widglag;
   actnet / act=task
            succ=succ
            lag=lagdur
            pcompress
            htext=3 boxht=3 arrowhead=2
            xbetween=7 ybetween=9
            centerid
            separatearcs;
   run;
```

Output 9.6.1 Network with Nonstandard Precedence Constraints

Widget Manufacture
Network Data Set

Obs	task	days	succ	lagdur
1	Approve Plan	5	Drawings	
2	Approve Plan	5	Study Market	
3	Approve Plan	5	Write Specs	
4	Drawings	10	Prototype	
5	Study Market	5	Mkt. Strat.	
6	Write Specs	5	Prototype	
7	Prototype	15	Materials	ss_9
8	Prototype	15	Facility	ss_9
9	Mkt. Strat.	10	Test Market	
10	Mkt. Strat.	10	Marketing	
11	Materials	10	Init. Prod.	
12	Facility	10	Init. Prod.	fs_2
13	Init. Prod.	10	Test Market	
14	Init. Prod.	10	Marketing	
15	Init. Prod.	10	Evaluate	
16	Evaluate	10	Changes	ss_6
17	Test Market	15	Changes	ff_3
18	Changes	5	Production	
19	Production	0		
20	Marketing	0		

Output 9.6.2 Network Diagram with Nonstandard Precedence Constraints

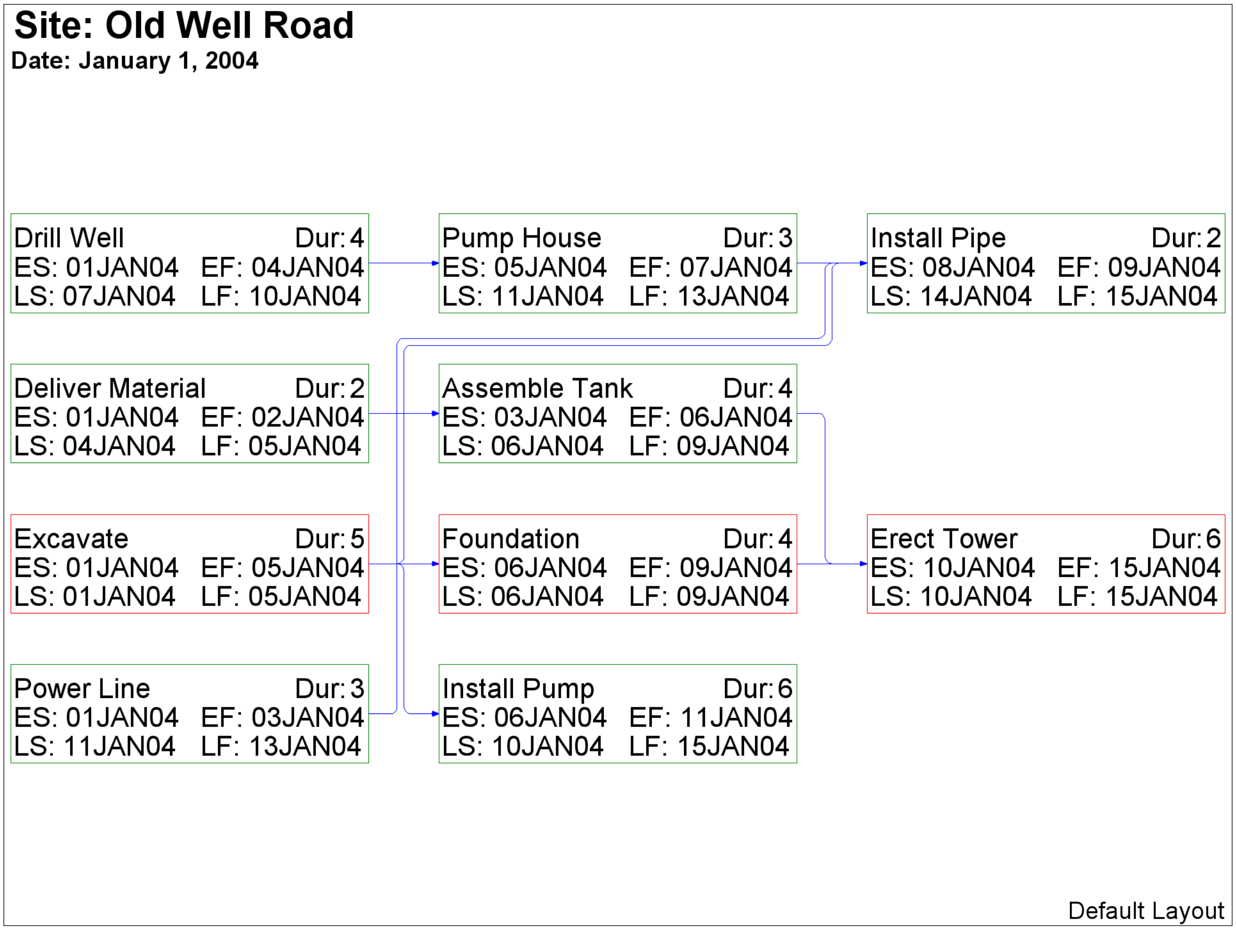


Example 9.7: Controlling the Arc-Routing Algorithm

This example illustrates the use of the DP and HTRACKS= options to control the routing of the arcs connecting the nodes. The project is a simple construction project with the following data. A Schedule data set produced by PROC CPM is input to PROC NETDRAW. The first invocation of the procedure illustrates the default layout of the network. As explained in the section "Layout of the Network" on page 690, the NETDRAW procedure uses a simple heuristic to route the arcs between the nodes. In the resulting diagram displayed in Output 9.7.1, note that the specification of BOXHT=3 limits the number of rows within each node so that the float values are not displayed.

```
data exmp1;
   input task     $ 1-16
         duration
         succesr1 $ 21-35
         succesr2 $ 36-50
         succesr3 $ 51-65;
   datalines;
Drill Well       4  Pump House
Pump House       3  Install Pipe
Power Line       3  Install Pipe
Excavate         5  Install Pipe   Install Pump   Foundation
Deliver Material 2  Assemble Tank
Assemble Tank    4  Erect Tower
Foundation       4  Erect Tower
Install Pump     6
Install Pipe     2
Erect Tower      6
;

proc cpm data=exmp1 date='1jan04'd out=sched;
   activity task;
   duration duration;
   successor succesr1 succesr2 succesr3;
   run;

pattern1 v=e c=green;
pattern2 v=e c=red;

title j=l h=3 ' Site: Old Well Road';
title2 j=l h=2 ' Date: January 1, 2004';
footnote j=r h=2 'Default Layout ';
proc netdraw data=sched graphics;
   actnet / act = task
            dur = duration
            succ = (succesr1-succesr3)
            boxht = 3 xbetween = 10
            separatearcs
            htext=2
            pcompress;
   run;
```

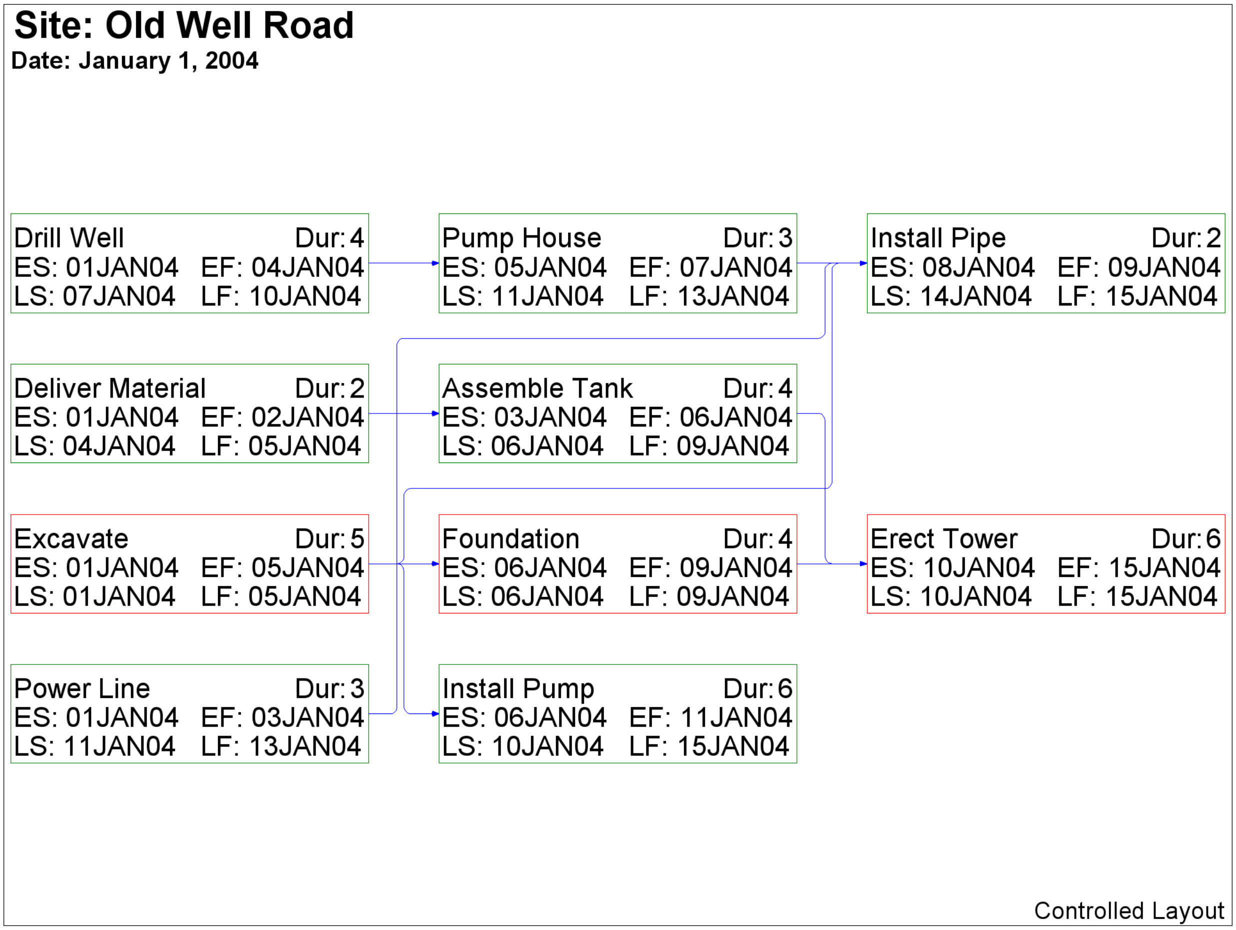
Output 9.7.1 Arc Routing: Default Layout



Next, a different routing of the arcs is obtained by specifying the DP and the HTRACKS= options. As a result of these options, the NETDRAW procedure uses a dynamic programming algorithm to route the arcs, limiting the number of horizontal tracks used to 1. The resulting network diagram is shown in Output 9.7.2. Notice that at most one arc is drawn in each horizontal track. Recall that, by default, the procedure uses a dynamic programming algorithm for arc routing if the number of tracks is restricted to be less than the maximum number of successors. Thus, for this example, the default routing option will be DP, even if it is not explicitly specified (because HTRACKS = 1 and the maximum number of successors is 3).

```
footnote j=r h=2 'Controlled Layout ';
proc netdraw data=sched graphics;
   actnet / act = task
            dur = duration
            succ = (succesr1-succesr3)
            boxht = 3 xbetween = 10
            separatearcs
            htracks=1
            htext=2
            pcompress
            dp;
   run;
```

Output 9.7.2 Arc Routing: Controlled Layout



Example 9.8: PATTERN and SHOWSTATUS Options

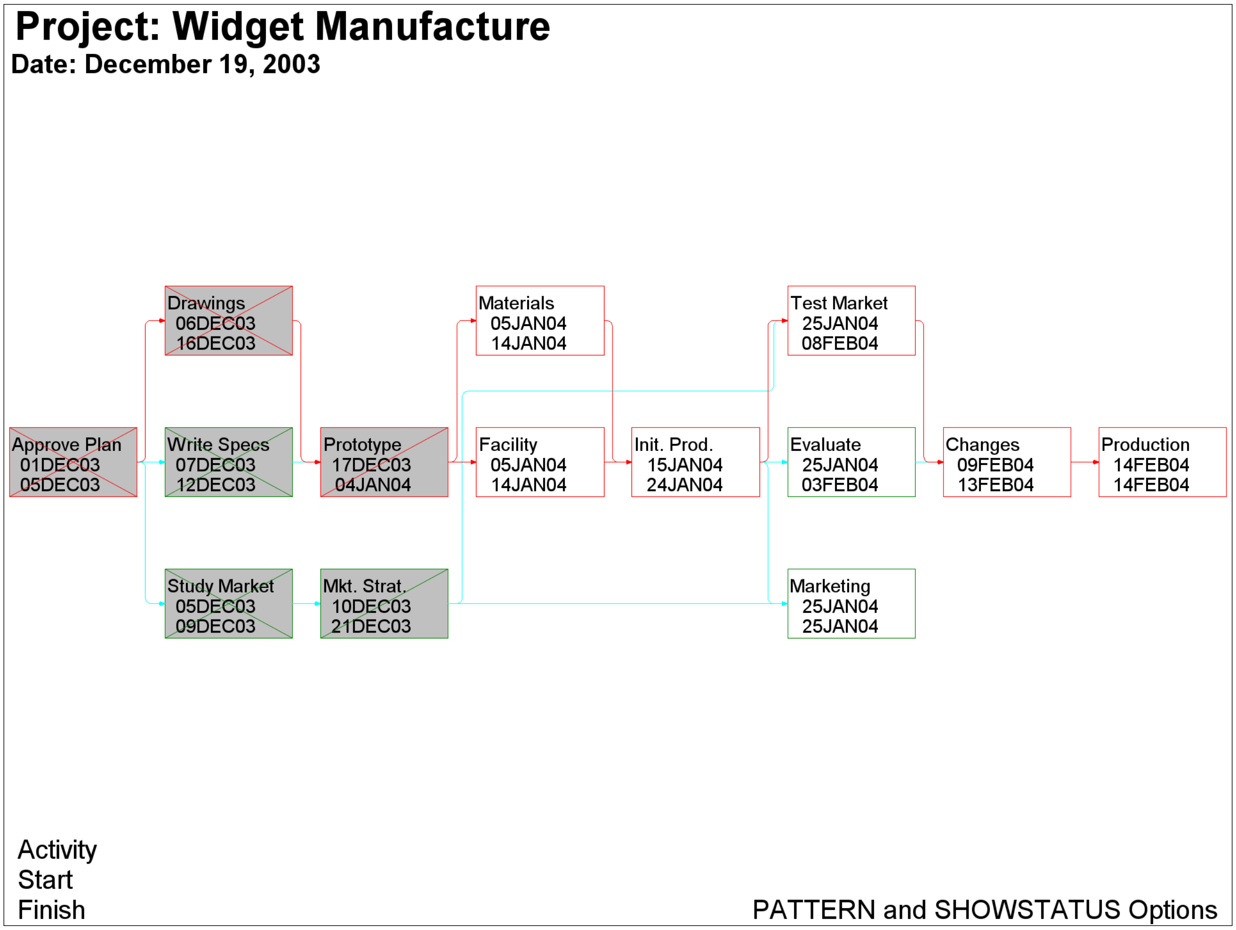
As a project progresses, in addition to the criticality of the activities, you may also want to display the status of the activity: whether it is in progress, has been completed, or is still to be scheduled. The SHOWSTATUS option in the ACTNET statement displays this additional information. In the current example, the same progress data as shown in Example 4.13 in Chapter 4, "The CPM Procedure," are used to illustrate the SHOWSTATUS option. The following program shows the necessary code. First, PROC CPM schedules the project with the SHOWFLOAT option; this enables activities that are already in progress or completed also to show nonzero float. Following this, a DATA step sets the variable style to '3' for activities that are completed or in progress; the remaining activities have missing values for this variable.

PROC NETDRAW is then invoked with the SHOWSTATUS option, which draws two diagonal lines across nodes referring to completed activities and one diagonal line for in-progress activities. The PATTERN= option in the ACTNET statement identifies the variable style containing the pattern information. Thus, the third pattern statement is used for in-progress or completed activities; the other activities (which have missing values for the variable style) use the second or the first pattern statement according to whether or not they are critical. However, since the first two PATTERN statements have EMPTY fill patterns specified, the nodes representing activities that have not yet started are in fact colored on the basis of the COUTLINE= and CCRITOUT= options. The resulting network diagram is shown in Output 9.8.1.

```
data holidays;
   format holiday holifin date7.;
   input holiday & date7. holifin & date7. holidur;
   datalines;
24dec03  26dec03  4
01jan04  .        .
;

* actual schedule at timenow = 19dec03;
data actual;
   format task $12. sdate fdate date7.;
   input task & sdate & date7. fdate & date7. pctc rdur;
   datalines;
Approve Plan  01dec03  05dec03  .    .
Drawings      06dec03  16dec03  .    .
Study Market  05dec03  .        100  .
Write Specs   07dec03  12dec03  .    .
Prototype     .        .        .    .
Mkt. Strat.   10dec03  .        .    3
Materials     .        .        .    .
Facility      .        .        .    .
Init. Prod.   .        .        .    .
Evaluate      .        .        .    .
Test Market   .        .        .    .
Changes       .        .        .    .
Production    .        .        .    .
Marketing     .        .        .    .
;

* merge the predicted information with network data;
data widgact;
   merge  actual widget;
   run;

* estimate schedule based on actual data;
proc cpm data=widgact holidata=holidays
         out=widgupd date='1dec03'd;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   actual / as=sdate af=fdate timenow='19dec03'd
            remdur=rdur pctcomp=pctc showfloat;
   run;

/* Set patterns for activities that have started */
data netin;
   set widgupd;
   if a_start ^= . then style = 3;
   run;
```

```
goptions hpos=120 vpos=70 border;
pattern1 c=green  v=e;
pattern2 c=red    v=e;
pattern3 c=ltgray v=s;

title  j=l h=3 ' Project: Widget Manufacture';
title2 j=l h=2 ' Date: December 19, 2003';
footnote1 j=l h=2 '  Activity';
footnote2 j=l h=2 '  Start';
footnote3 j=l h=2 '  Finish'
          j=r h=2 'PATTERN and SHOWSTATUS Options  ';
proc netdraw data=netin graphics;
   actnet / act=task
            succ=(succ1 succ2 succ3)
            ybetween = 10
            separatearcs
            pcompress
            id=(task e_start e_finish)
            nodefid nolabel
            carcs=cyan
            ccritarcs=red
            coutline = green
            ccritout = red
            showstatus
            pattern = style
            htext=2;
   run;
```

Output 9.8.1 PATTERN and SHOWSTATUS Options



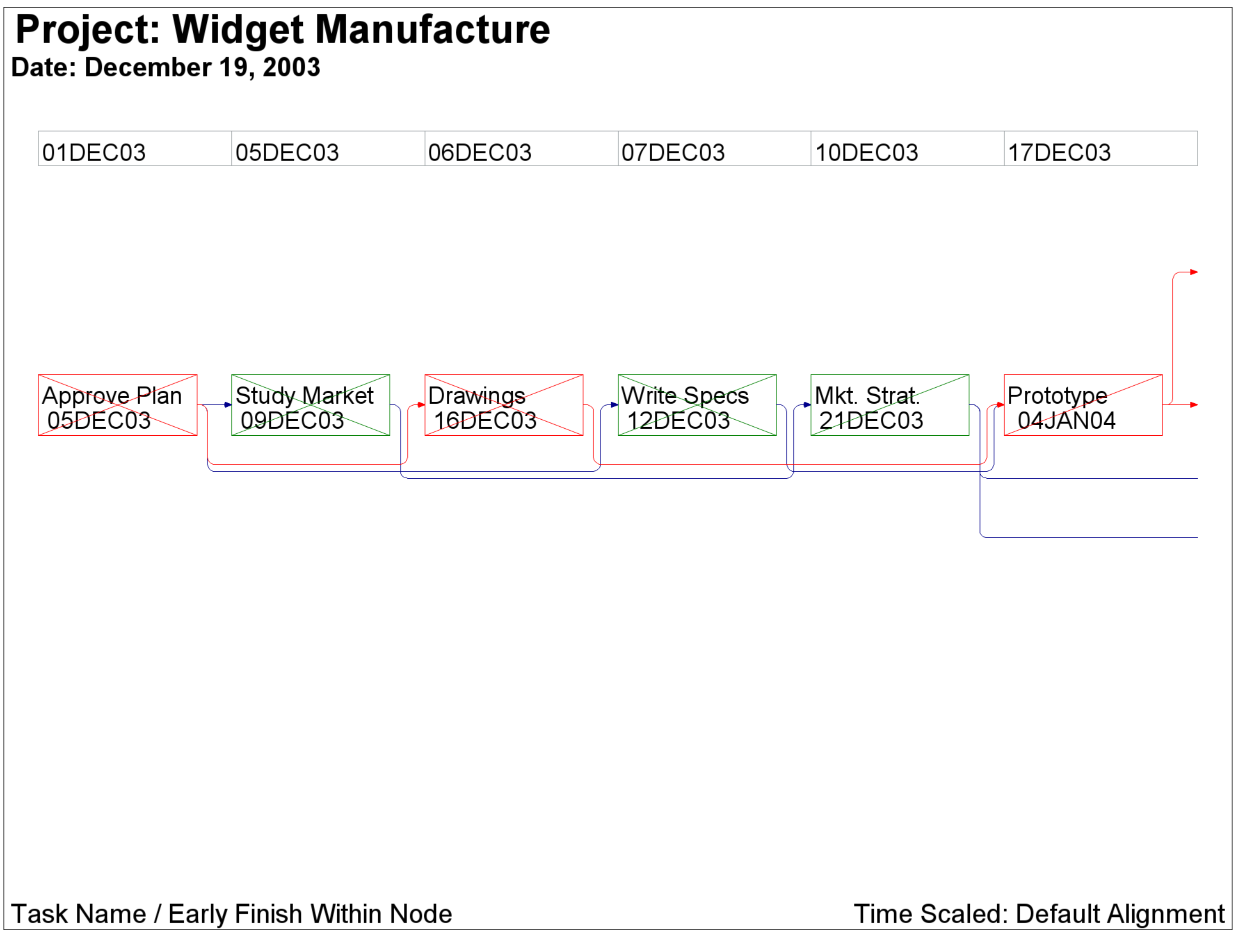
Example 9.9: Time-Scaled Network Diagram

This example illustrates the use of the TIMESCALE and ALIGN= options to draw time-scaled network diagrams. The Schedule data set WIDGUPD, produced by PROC CPM in the previous example, is used. First, PROC NETDRAW is invoked with the TIMESCALE option without any ALIGN= specification. By default, the procedure aligns the nodes to coincide with the early start times of the activities. The network spans two pages (HPAGES=2 VPAGES=1), as shown in Output 9.9.1. The HMARGIN= and VMARGIN= options add extra space around the margins.

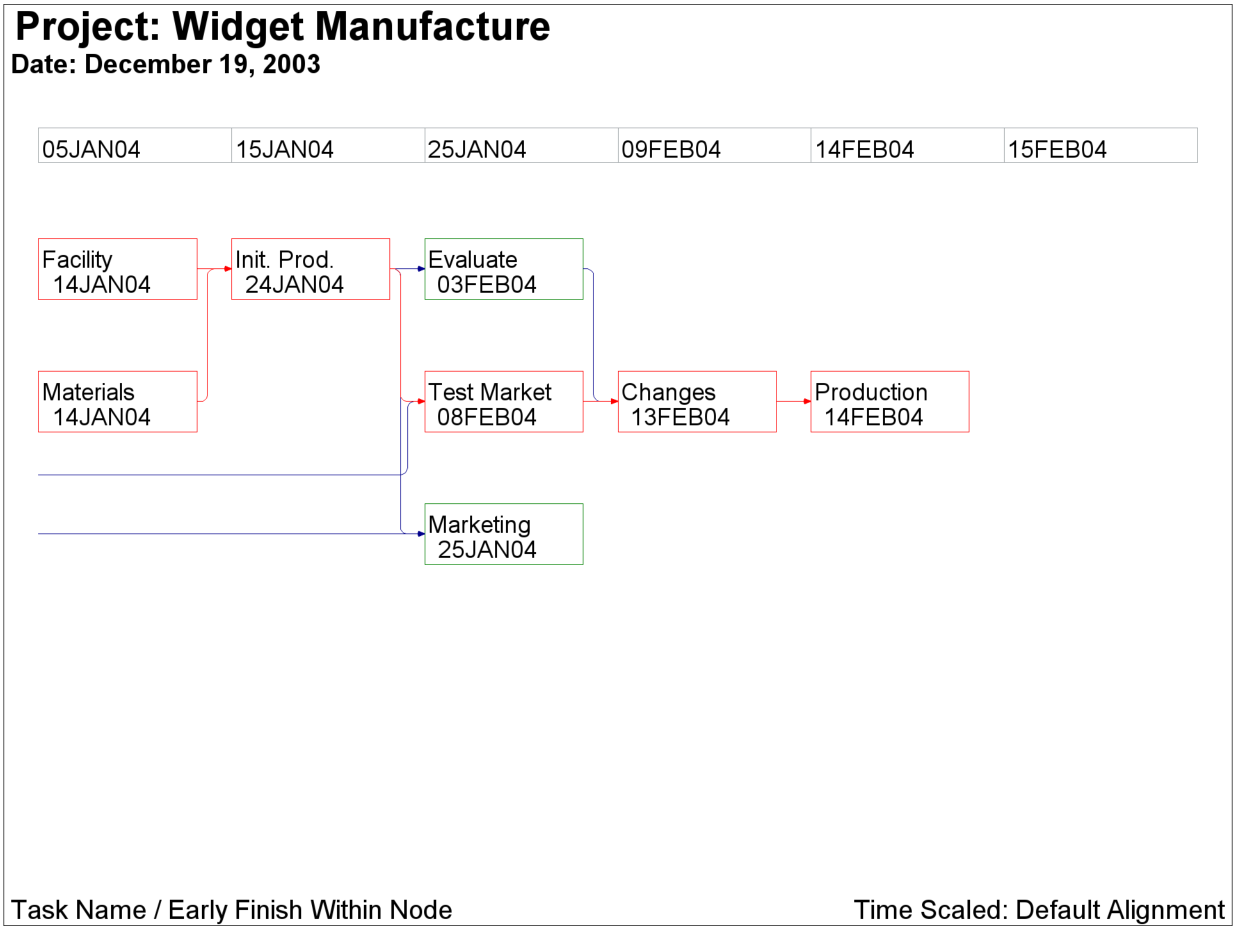
```
pattern1 v=e c=green;
pattern2 v=e c=red;

title  j=l h=3 ' Project: Widget Manufacture';
title2 j=l h=2 ' Date: December 19, 2003';
footnote j=l h=2 ' Task Name / Early Finish Within Node'
         j=r h=2 'Time Scaled: Default Alignment ';
proc netdraw data=widgupd graphics;
   actnet / act=task
            succ=(succ1 succ2 succ3)
            ybetween = 8
```

```
            separatearcs
            novcenter
            id=(task e_finish) nodefid
            nolabel
            showstatus
            carcs=darkblue
            ccritarcs=red
            vmargin=5
            hmargin=5
            timescale
            htext=2  pcompress
            hpages=2 vpages=1
            nopagenumber;
run;
```

Output 9.9.1 TIMESCALE Option: Default Alignment



Output 9.9.1 *continued*



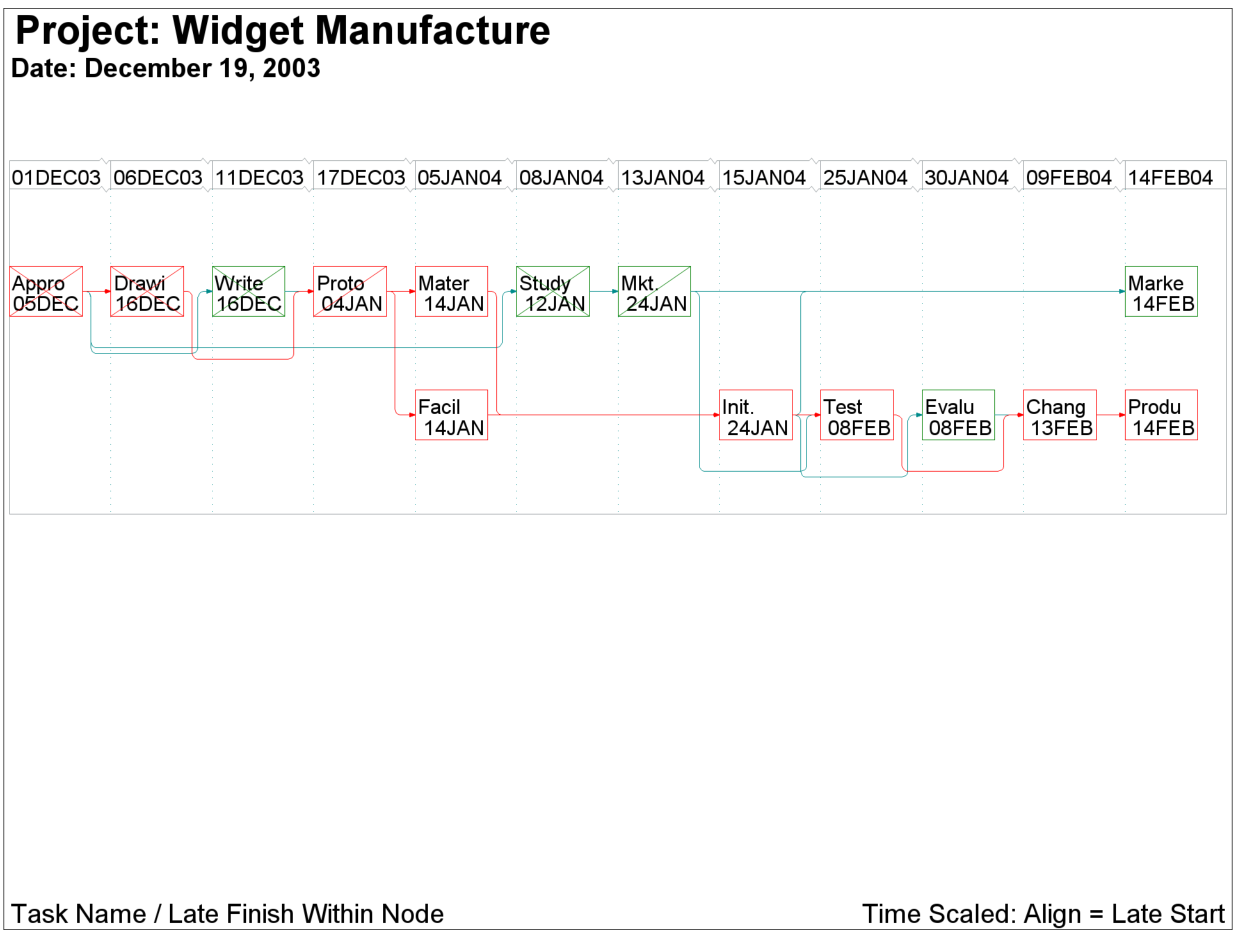
Next, PROC NETDRAW is invoked with several of the time-scale options:

- The ALIGN= option requests that the activities be placed according to the L_START times.
- The FRAME option produces a border around the network diagram.
- The AUTOREF option draws reference lines at every tick mark.
- The LREF= and CREF= options specify the line style and color for the reference lines.
- The SHOWBREAK option requests that breaks in the time axis be indicated by breaks before the corresponding tick marks.

```
footnote j=l h=2 ' Task Name / Late Finish Within Node'
         j=r h=2 'Time Scaled: Align = Late Start ';
proc netdraw data=widgupd graphics;
   actnet / act=task
            succ=(succ1 succ2 succ3)
            ybetween = 10
            separatearcs
```

```
            pcompress
            novcenter
            id=(task l_finish) nodefid
            nolabel
            boxwidth=5
            showstatus
            carcs=darkcyan
            ccritarcs=red
            vmargin=10
            align=l_start
            frame
            autoref
            lref=33
            cref=darkcyan
            showbreak
            htext=2;
run;
```

Output 9.9.2 Timescale Option: ALIGN= L_START



Example 9.10: Further Time-Scale Options

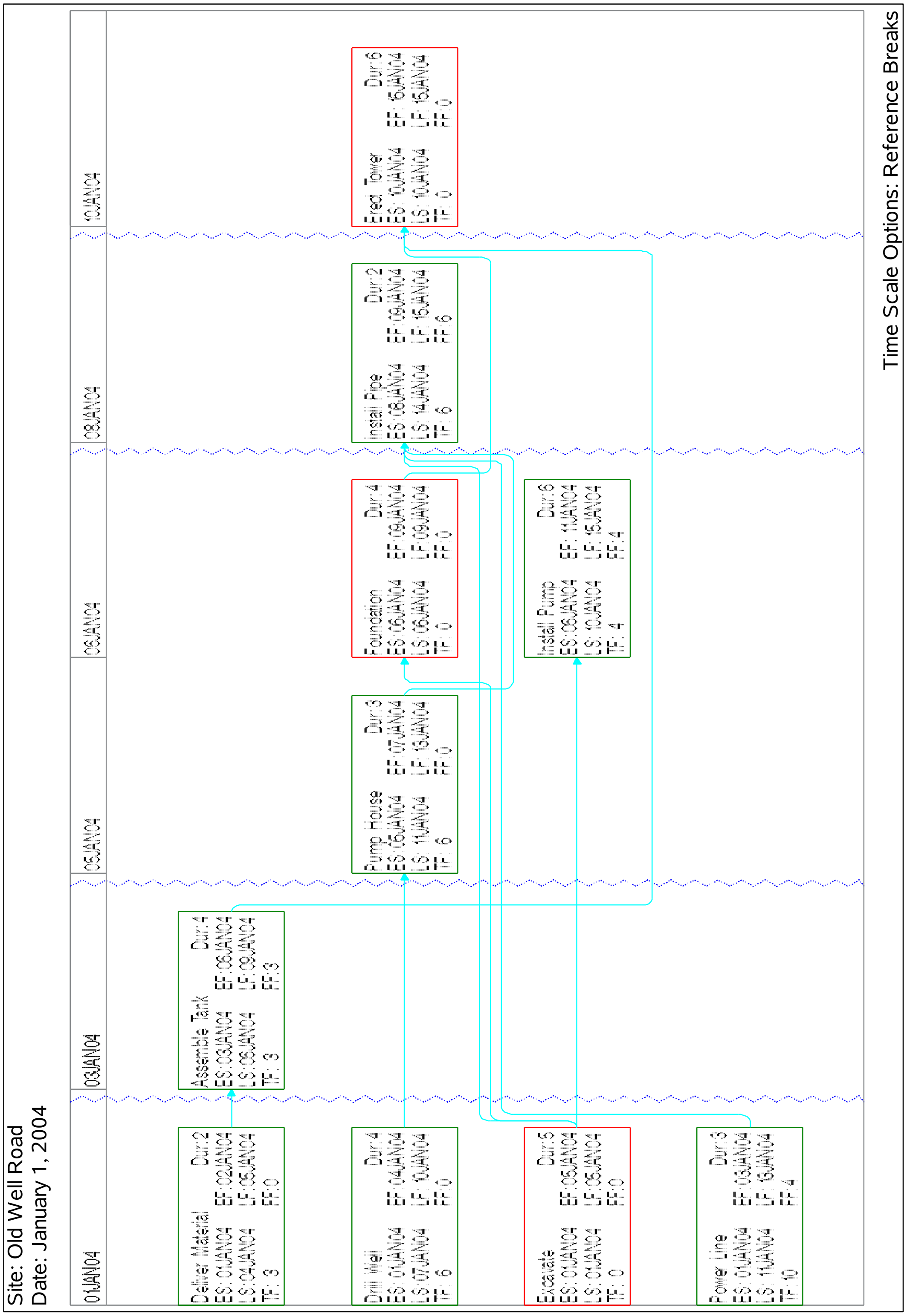
In this example, the construction project described in Example 9.7 is used to illustrate some more time-scale options. First, the REFBREAK option indicates breaks in the time axis by drawing a zigzag line down the diagram before each tick mark corresponding to a break. The CREFBRK= and LREFBRK= options control the color and line style for these lines. The network diagram is shown in Output 9.10.1.

```
title j=l h=1.5 ' Site: Old Well Road';
title2 j=l h=1.5 ' Date: January 1, 2004';
footnote j=r h=1.5 'Time Scale Options: Reference Breaks ';


proc netdraw data=sched graphics;
   actnet / act = task
            dur = duration
            succ = (succesr1-succesr3)
            dp compress separatearcs
            font=swiss htext=2
            timescale refbreak lrefbrk = 33
            carcs=cyan crefbrk = blue;
   run;
```

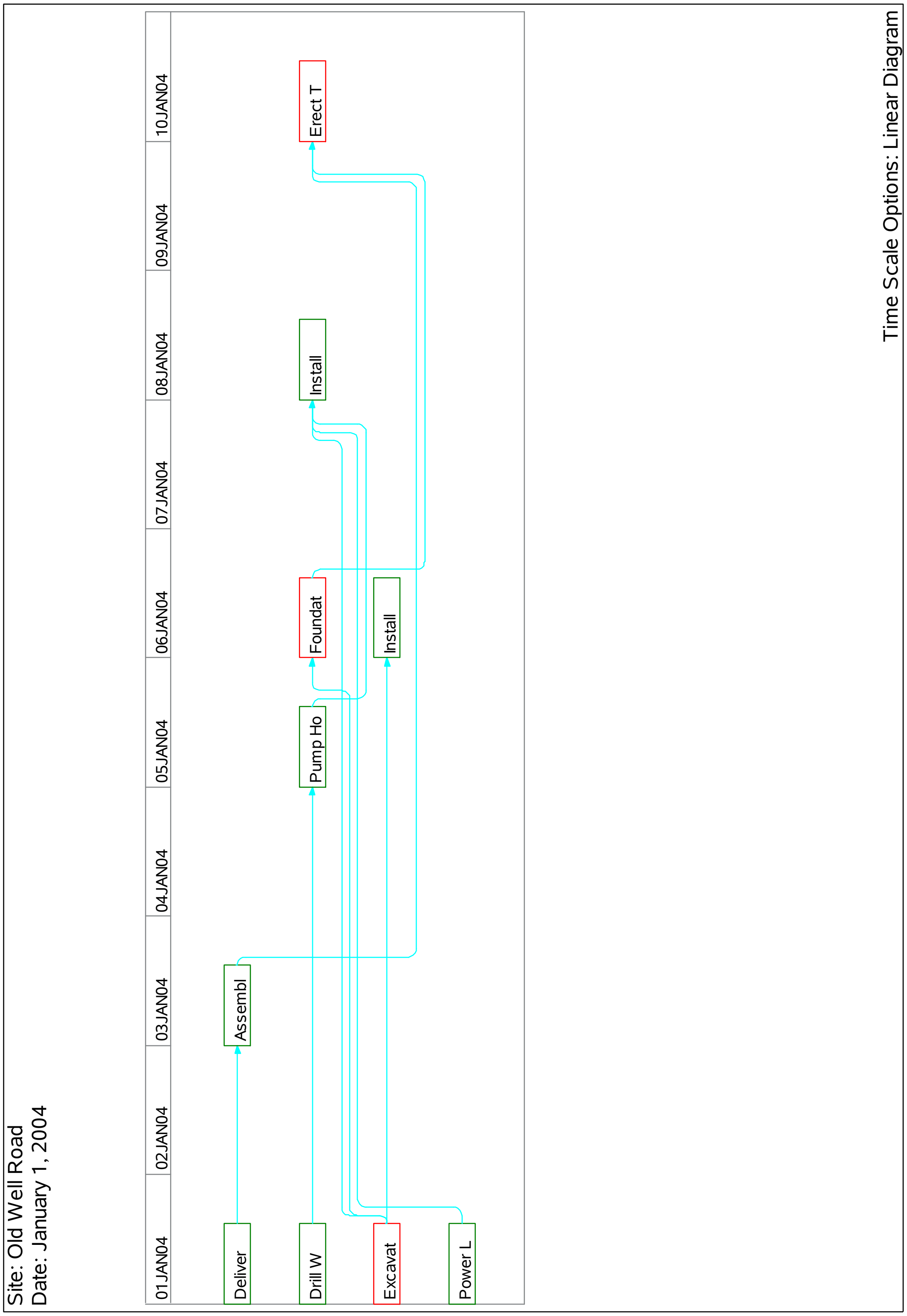
Next, PROC NETDRAW is invoked with the LINEAR option so that there is no break in the time axis. The BOXWIDTH= option limits the size of each node. The diagram is drawn in Output 9.10.2.

```
title j=l h=1.5 ' Site: Old Well Road';
title2 j=l h=1.5 ' Date: January 1, 2004';
footnote j=r h=1.5 'Time Scale Options: Linear Diagram ';

proc netdraw data=sched graphics;
   actnet / act = task
            dur = duration
            succ = (succesr1-succesr3)
            dp
            pcompress
            novcenter
            vmargin = 10
            separatearcs
            htext=2
            carcs=cyan
            id=(task)
            nodefid
            nolabel
            boxwidth=7
            timescale
            linear
            frame;
   run;
```

Output 9.10.1 Time-Scaled Network: Reference Breaks



Output 9.10.2 Time-Scaled Network: LINEAR Option



Example 9.11: Zoned Network Diagram

This example illustrates zoned network diagrams. The Widget Manufacturing project is used to illustrate some aspects of this feature. The data set DETAILS contains a variable phase, which identifies the phase of each activity in the project. This data set is merged with the Activity data set from Example 9.1, WIDGET, to produce the data set NETWORK that is input to PROC NETDRAW. The ZONE= option divides the network diagram into horizontal zones based on the project phase. The ZONEPAT option causes the activities in each zone to be drawn using a different pattern. The resulting network diagram is shown in Output 9.11.1.

```
data details;
   format task $12. phase $13. descrpt $30. ;
   input task & phase $ descrpt & ;
   datalines;
Approve Plan  Planning       Develop Concept
Drawings      Engineering    Prepare Drawings
Study Market  Marketing      Analyze Potential Markets
Write Specs   Engineering    Write Specifications
Prototype     Engineering    Build Prototype
Mkt. Strat.   Marketing      Develop Marketing Concept
Materials     Manufacturing  Procure Raw Materials
Facility      Manufacturing  Prepare Manufacturing Facility
Init. Prod.   Manufacturing  Initial Production Run
Evaluate      Testing        Evaluate Product In-House
Test Market   Testing        Test Product in Sample Market
Changes       Engineering    Engineering Changes
Production    Manufacturing  Begin Full Scale Production
Marketing     Marketing      Begin Full Scale Marketing
;
data network;
   merge widget details;
   run;

pattern1 v=e c=green;
pattern2 v=e c=red;
pattern3 v=e c=magenta;
pattern4 v=e c=blue;
pattern5 v=e c=cyan;

title  j=l h=1.5 ' Project: Widget Manufacture';
title2 j=l h=1.5 ' Date: December 1, 2003';
footnote j=r h=1.5 'Zoned Network Diagram ';


proc netdraw data=network graphics;
   actnet / act=task succ=(succ1 succ2 succ3)
            separatearcs
            zone=phase
            zonepat
            pcompress
            htext=2;
   label phase = 'Department';
   run;
```

Next, the project is scheduled with PROC CPM, and PROC NETDRAW is invoked with the ZONE= and TIMESCALE options. The nodes are placed in different zones as dictated by the ZONE variable, phase, and are aligned along the time axis as dictated by the default ALIGN variable, E_START. The MININTERVAL= option produces one tick mark per week for the duration of the project. The LREF= option identifies the linestyle of the reference lines and the dividing lines between zones. The nodes are colored red or green according to whether or not the corresponding activities are critical (PATTERN statements 1 and 2 from the previous invocation of PROC NETDRAW are still valid).

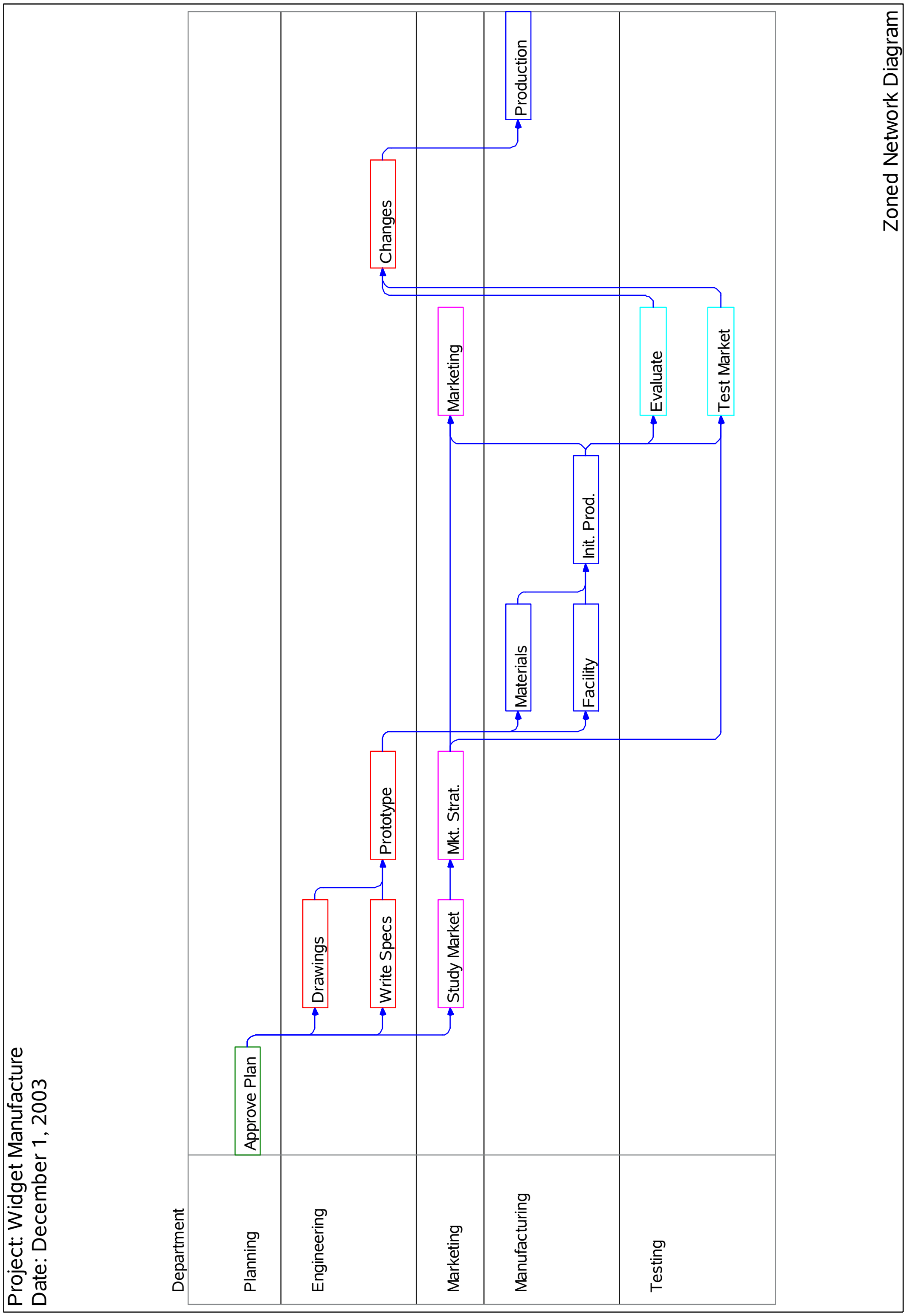
```
proc cpm data=network interval=weekday
         out=sched date='1dec03'd;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   id phase;
   run;

title  j=l h=1.5 ' Project: Widget Manufacture';
title2 j=l h=1.5 ' Date: December 1, 2003';
footnote j=r h=1.5 'Zone and Timescale ';
proc netdraw data=sched graphics;
   actnet / act=task succ=(succ1 succ2 succ3)
            pcompress
            carcs = blue ccritarcs=red
            cref = cyan
            caxis = magenta
            lref = 33
            id = (task)
            nodefid
            nolabel
            boxwidth = 8
            htext=2
            separatearcs
            timescale
            mininterval=week
            autoref
            linear
            zone=phase
            zonespace;
   label phase = 'Department';
   run;
```

Example 9.12: Schematic Diagrams

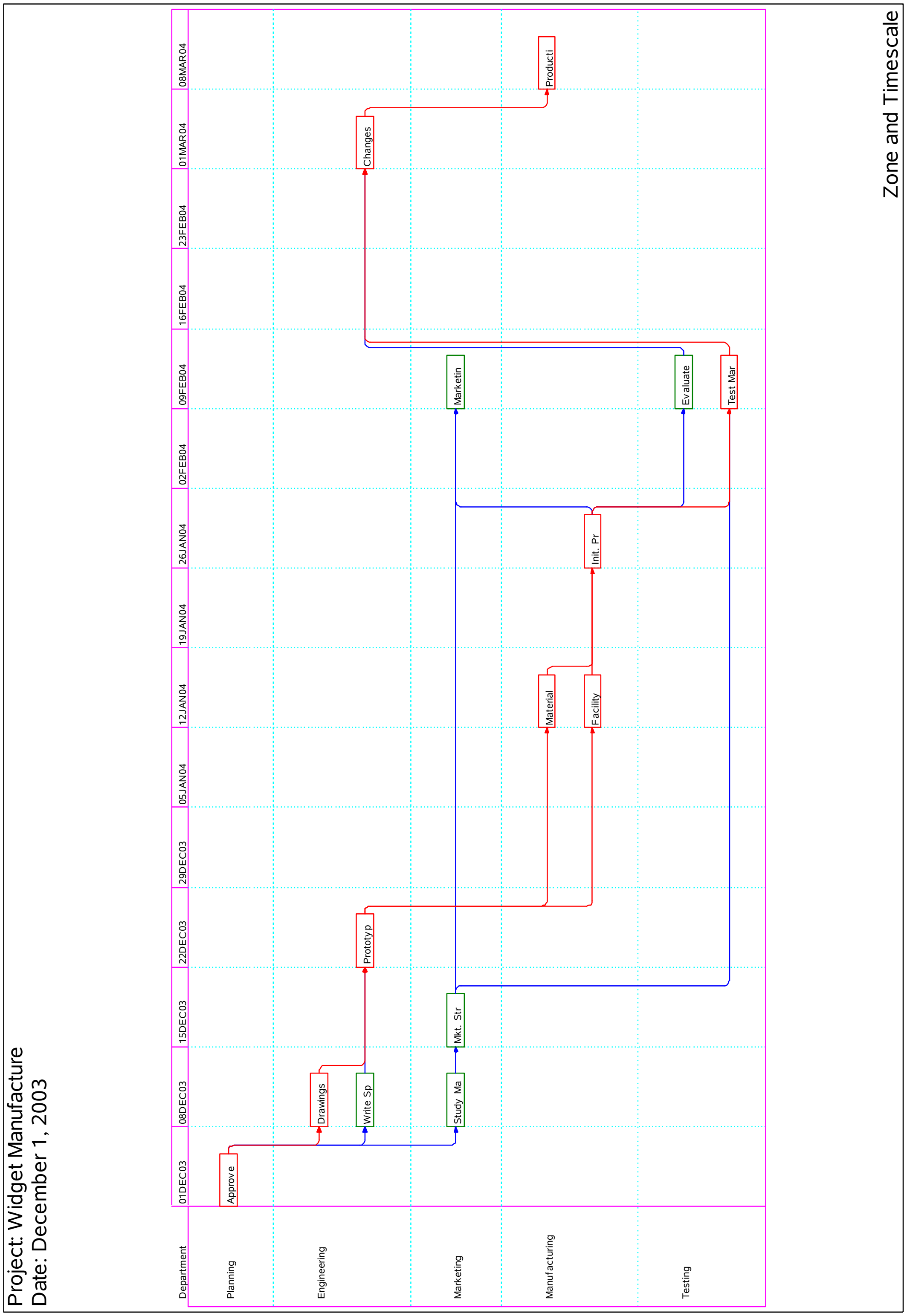
You can use PROC NETDRAW to determine node placement and arc routing for any network depicting a set of nodes connected by arcs. If you want the procedure to determine the node placement, the network must be acyclic. This example illustrates the use of PROC NETDRAW to draw two networks that represent different schematic flows. The first network does not contain any cycles, while the second one has one cycle; to draw the second network, you need to use the BREAKCYCLE option.

First, a schematic representation of the data flow going in and out of the three procedures (CPM, GANTT, and NETDRAW) is drawn using PROC NETDRAW. (See Chapter 3, "Introduction to Project Management,"

Output 9.11.1 Zoned Network Diagram



Output 9.11.2 Zoned Network Diagram with Time Axis

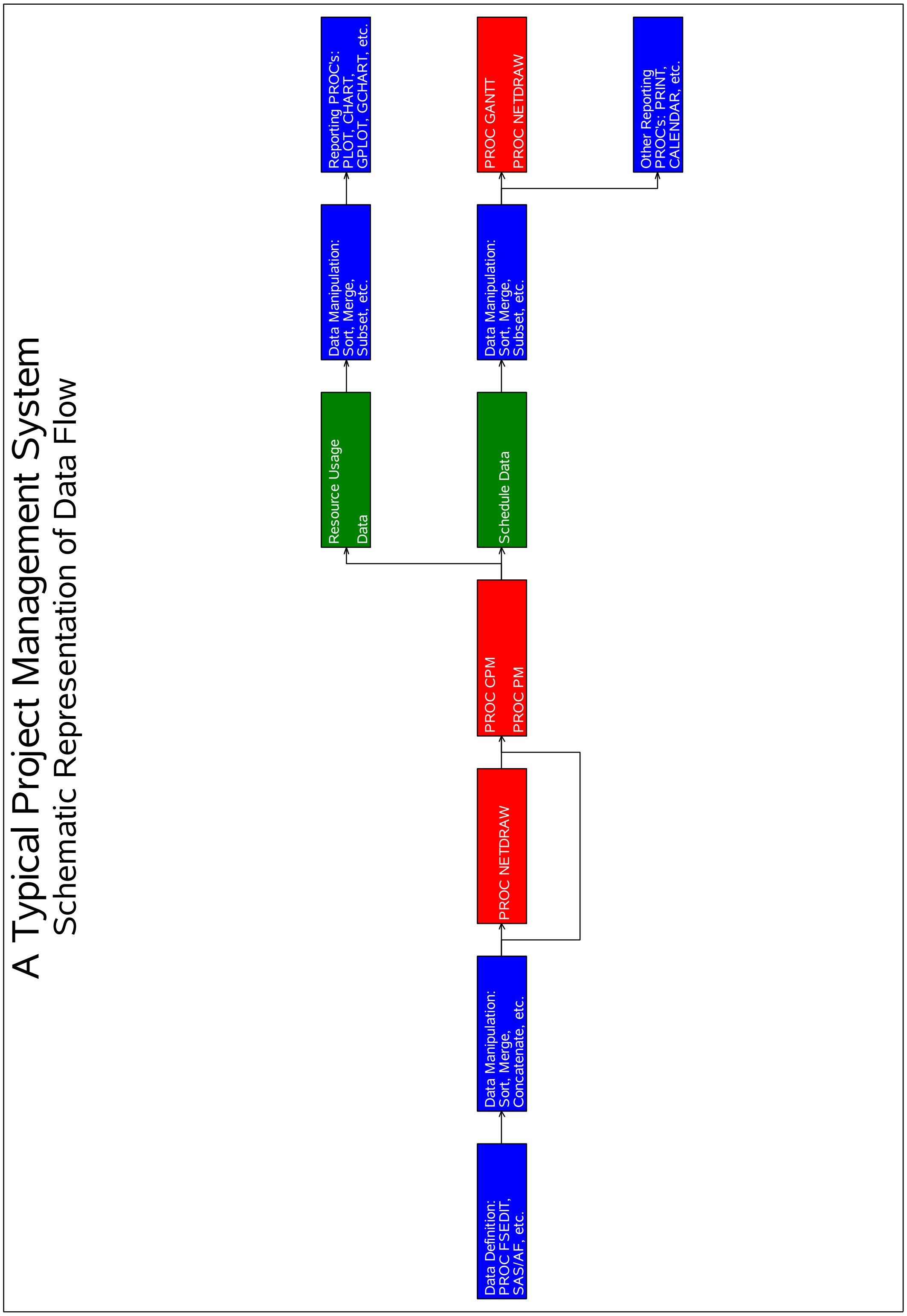


for a detailed discussion of such a data flow.) The PATTERN= option is used to specify the variable in the data set that identifies the color that is to be used for each node. Nodes representing SAS/OR procedures are colored red, the ones representing output data sets are colored green, and all other nodes (representing the use of other parts of the SAS System) are colored blue. Three ID variables are used to specify the text that is to be placed within each node. The flow diagram is shown in Output 9.12.1.

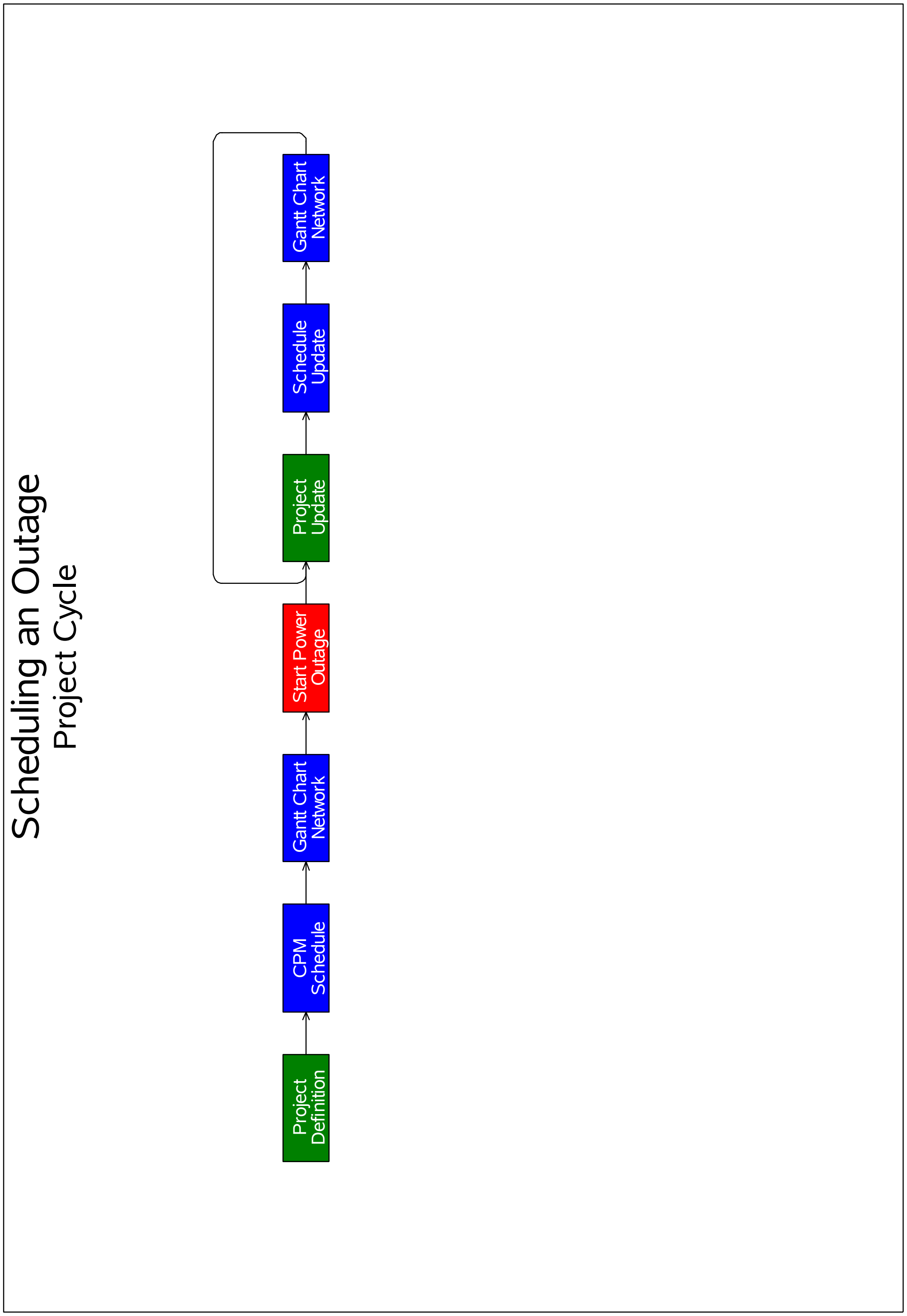
```
data dataflow;
   format id1 $18. id2 $14. id3 $19. ;
   input a $ b $ id1 & id2 & id3 & style;
   datalines;
A B Data Definition:    PROC FSEDIT,    SAS/AF, etc.        2
B C Data Manipulation:  Sort, Merge,    Concatenate, etc.   2
B D Data Manipulation:  Sort, Merge,    Concatenate, etc.   2
D C .                   PROC NETDRAW    .                   1
C E PROC CPM            .               PROC PM             1
C F PROC CPM            .               PROC PM             1
E H Resource Usage      .               Data                3
F G .                   Schedule Data   .                   3
G I Data Manipulation:  Sort, Merge,    Subset, etc.        2
G J Data Manipulation:  Sort, Merge,    Subset, etc.        2
H K Data Manipulation:  Sort, Merge,    Subset, etc.        2
I . Other Reporting     PROC's: PRINT,  CALENDAR, etc.      2
J . PROC GANTT          .               PROC NETDRAW        1
K . Reporting PROC's:   PLOT, CHART,    GPLOT, GCHART, etc. 2
;

pattern1 v=s c=red;
pattern2 v=s c=blue;
pattern3 v=s c=green;

goptions hpos=110 vpos=70;
title h=3 'A Typical Project Management System';
title2 h=2.5  'Schematic Representation of Data Flow';
proc netdraw data=dataflow graphics;
   actnet / act=a succ=b id = (id1-id3)
            nodefaultid
            nolabel
            pattern=style
            carcs=black coutline=black ctext=white
            hmargin = 2
            ybetween = 15
            rectilinear
            noarrowfill
            pcompress htext=2;
   run;
```

Next, a typical sequence of procedures followed at the scheduling of a nuclear power plant outage is shown using the NETDRAW procedure. Such a schematic diagram is illustrated in Chapter 3, "Introduction to Project Management." In Figure 3.6, there is a cycle that is not normally allowed in a Network data set that is input to PROC NETDRAW. However, you can draw such networks by specifying the BREAKCYCLE option. (Note that you can also draw cyclic networks by specifying explicitly the node coordinates or an ALIGN= variable that fixes the *x* coordinates for each node.)

Output 9.12.1 Schematic Representation of Data Flow



Output 9.12.2 Scheduling a Power Outage



In this example, the data set OUTAGE contains the network representation. The variable style is used to color nodes appropriately. The resulting diagram is shown in Output 9.12.2.

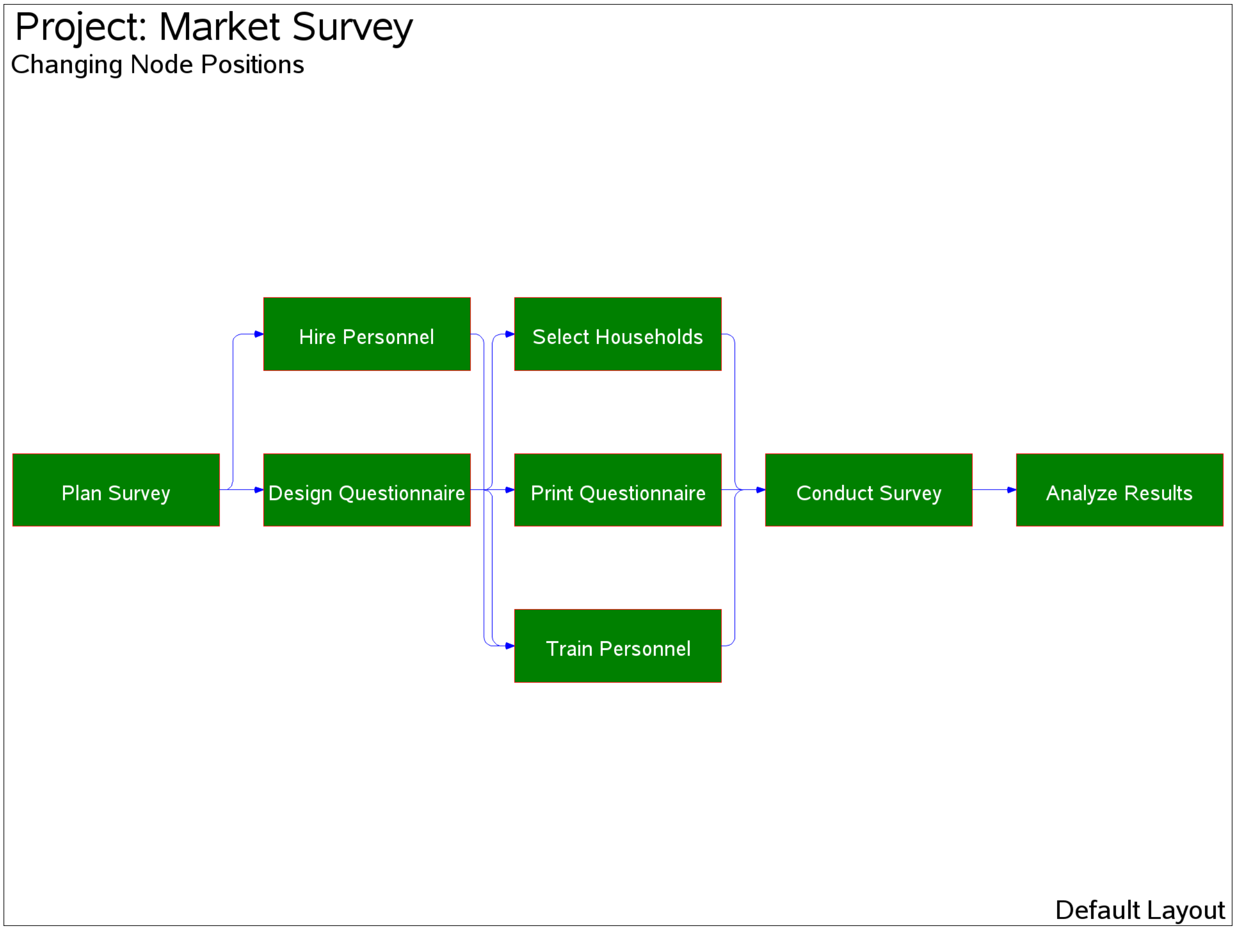
```
data outage;
   input a $ b $ id1 $20.  id2 $20. style;
   datalines;
A   B   Project            Definition            1
B   C   CPM                Schedule              2
C   D   Gantt Chart        Network               3
D   E   Start Power        Outage                4
E   F   Project            Update                1
F   G   Schedule           Update                2
G   E   Gantt Chart        Network               3
;

goptions hpos=110 vpos=70;
title h=3 'Scheduling an Outage';
title2 h=2.5  'Project Cycle';

pattern1 v=s c=green;
pattern2 v=s c=blue;
pattern3 v=s c=blue;
pattern4 v=s c=red;

proc netdraw data=outage graphics;
   actnet / act=a succ=b id = (id1 id2)
            breakcycle
            nodefaultid centerid
            vmargin = 5 hmargin = 0
            nolabel novcenter
            pattern=style
            carcs=black coutline=black ctext=white
            ybetween = 15 xbetween=3
            noarrowfill
            pcompress htext=2;
   run;
```

Example 9.13: Modifying Network Layout

This example uses the SURVEY project described in Chapter 3, “Introduction to Project Management,” to illustrate how you can modify the default layout of the network. The data set SURVEY contains the project information. PROC NETDRAW is invoked with the GRAPHICS option. The network diagram is shown in Output 9.13.1.

```
data survey;
   format id $20. activity succ1-succ3 $8. phase $9. ;
   input id        &
         activity  &
         duration
         succ1     &
         succ2     &
         succ3     &
         phase     $ ;
   datalines;
Plan Survey           plan sur   4 hire per  design q  .        Plan
Hire Personnel        hire per   5 trn per   .         .        Prepare
Design Questionnaire  design q   3 trn per   select h  print q  Plan
Train Personnel       trn per    3 cond sur  .         .        Prepare
Select Households     select h   3 cond sur  .         .        Prepare
Print Questionnaire   print q    4 cond sur  .         .        Prepare
Conduct Survey        cond sur  10 analyze   .         .        Implement
Analyze Results       analyze    6 .         .         .        Implement
;


pattern1 v=s c=green;

title j=l h=3' Project: Market Survey';
title2 j=l h=2 ' Changing Node Positions';
footnote j=r h=2 'Default Layout ';

proc netdraw data=survey graphics out=network;
   actnet / act=activity
            succ=(succ1-succ3)
            id=(id) nodefid nolabel
            carcs = blue
            ctext  = white
            coutline=red
            centerid
            boxht = 3
            htext=2
            pcompress
            separatearcs
            ybetween=8;
   run;

footnote;
title2 'NETWORK Output Data Set';
proc print data=network;
   run;
```

Output 9.13.1 Default Network Layout of SURVEY Project



The Layout data set produced by PROC NETDRAW (displayed in Output 9.13.2) contains the *x* and *y* coordinates for all the nodes in the network and for all the turning points of the arcs connecting them.

Suppose that you want to interchange the positions of the nodes corresponding to the two activities, 'Select Households' and 'Train Personnel.' As explained in the section "Controlling the Layout" on page 695, you can invoke the procedure in FULLSCREEN mode and use the MOVE command to move the nodes to desired locations. In this example, the data set NETWORK produced by PROC NETDRAW is used to change the *x* and *y* coordinates of the nodes. A new data set called NODEPOS is created from NETWORK by retaining only the observations containing node positions (recall that for such observations, _SEQ_ = '0') and by dropping the _SEQ_ variable. Further, the *y* coordinates (given by the values of the _Y_ variable) for the two activities 'Select Households' and 'Train Personnel' are interchanged. The new data set, displayed in Output 9.13.3, is then input to PROC NETDRAW.

Output 9.13.2 Layout Data Set

Project: Market Survey

NETWORK Output Data Set

Obs	_FROM_	_TO_	_X_	_Y_	_SEQ_	_PATTERN	id
1	plan sur	hire per	1.0	2	0	1	Plan Survey
2	plan sur	hire per	1.5	2	1	.	Plan Survey
3	plan sur	hire per	1.5	3	2	.	Plan Survey
4	plan sur	design q	1.0	2	0	1	Plan Survey
5	hire per	trn per	2.0	3	0	1	Hire Personnel
6	hire per	trn per	2.5	3	1	.	Hire Personnel
7	hire per	trn per	2.5	1	2	.	Hire Personnel
8	design q	trn per	2.0	2	0	1	Design Questionnaire
9	design q	trn per	2.5	2	1	.	Design Questionnaire
10	design q	trn per	2.5	1	2	.	Design Questionnaire
11	design q	select h	2.0	2	0	1	Design Questionnaire
12	design q	select h	2.5	2	1	.	Design Questionnaire
13	design q	select h	2.5	3	2	.	Design Questionnaire
14	design q	print q	2.0	2	0	1	Design Questionnaire
15	trn per	cond sur	3.0	1	0	1	Train Personnel
16	trn per	cond sur	3.5	1	1	.	Train Personnel
17	trn per	cond sur	3.5	2	2	.	Train Personnel
18	select h	cond sur	3.0	3	0	1	Select Households
19	select h	cond sur	3.5	3	1	.	Select Households
20	select h	cond sur	3.5	2	2	.	Select Households
21	print q	cond sur	3.0	2	0	1	Print Questionnaire
22	cond sur	analyze	4.0	2	0	1	Conduct Survey
23	analyze		5.0	2	0	1	Analyze Results

```
data nodepos;
   set network;
   if _seq_ = 0;
   drop _seq_;
   if _from_ = 'select h' then _y_=1;
   if _from_ = 'trn per' then _y_=3;
   run;

title2 'Modified Node Positions';
proc print data=nodepos;
   run;
```

Output 9.13.3 Modified Layout Data Set

Project: Market Survey

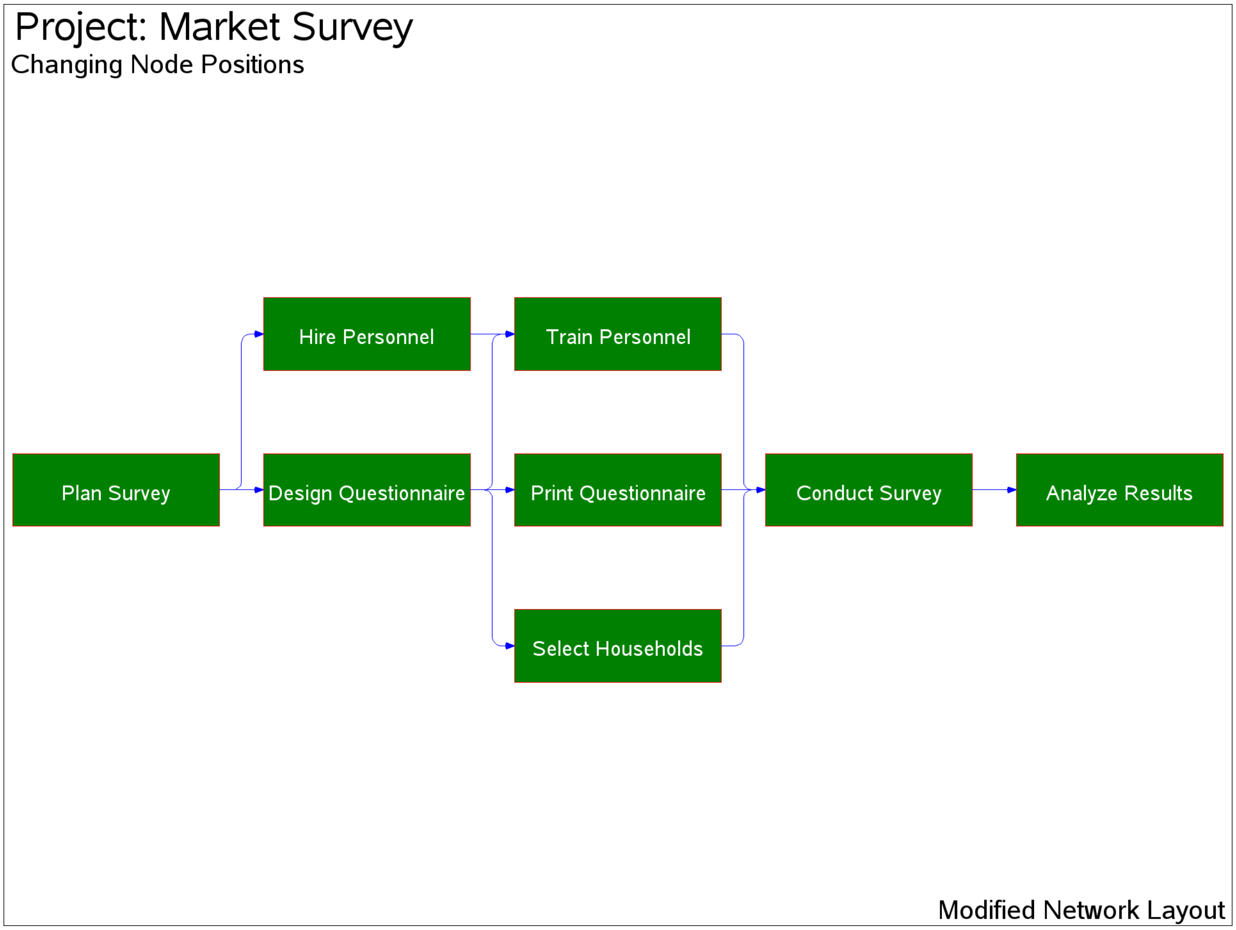
Modified Node Positions

Obs	_FROM_	_TO_	_X_	_Y_	_PATTERN	id
1	plan sur	hire per	1	2	1	Plan Survey
2	plan sur	design q	1	2	1	Plan Survey
3	hire per	trn per	2	3	1	Hire Personnel
4	design q	trn per	2	2	1	Design Questionnaire
5	design q	select h	2	2	1	Design Questionnaire
6	design q	print q	2	2	1	Design Questionnaire
7	trn per	cond sur	3	3	1	Train Personnel
8	select h	cond sur	3	1	1	Select Households
9	print q	cond sur	3	2	1	Print Questionnaire
10	cond sur	analyze	4	2	1	Conduct Survey
11	analyze		5	2	1	Analyze Results

Note that the data set NODEPOS contains variables named _FROM_ and _TO_, which specify the (activity, successor) information; hence, the call to PROC NETDRAW does not contain the ACTIVITY= and SUCCESSOR= specifications. The presence of the variables _X_ and _Y_ indicates to PROC NETDRAW that the data set contains the *x* and *y* coordinates for all the nodes. Because there is no variable named _SEQ_ in this data set, PROC NETDRAW assumes that only the node coordinates are given and uses these node positions to determine how the arcs are to be routed. The resulting network diagram is shown in Output 9.13.4.

```
title j=l h=3 ' Project: Market Survey';
title2 j=l h=2 ' Changing Node Positions';
footnote j=r h=2 'Modified Network Layout ';

proc netdraw data=nodepos graphics;
   actnet / id=(id) nodefid nolabel
            carcs = blue
            ctext = white
            coutline = red
            centerid
            boxht = 3
            htext=2
            pcompress
            separatearcs
            ybetween=8;
   run;
```

Output 9.13.4 Modified Network Layout of SURVEY Project



Example 9.14: Specifying Node Positions

This example uses a typical problem in network flow optimization to illustrate how you can use PROC NETDRAW to draw a network by specifying completely all the node positions. Consider a simple two-period production inventory problem with one manufacturing plant (PLANT), two warehouses (DEPOT1 and DEPOT2), and one customer (CUST). In each period, the customer can receive goods directly from the plant or from the two warehouses. The goods produced at the plant can be used to satisfy directly some or all of the customer's demands or can be shipped to a warehouse. Some of the goods can also be carried over to the next period as inventory at the plant. The problem is to determine the minimum cost of satisfying the customer's demands; in particular, how much of the customer's demands in each period is to be satisfied from the inventory at the two warehouses or from the plant, and also how much of the production is to be carried over as inventory at the plant? This problem can be solved using PROC NETFLOW; the details are not discussed here. Let PLANT_**`i`** represent the production at the plant in period **`i`**, DEPOT1_**`i`** represent the inventory at DEPOT1 in period **`i`**, DEPOT2_**`i`** represent the inventory at DEPOT2 in period **`i`**, and CUST_**`i`** represent the customer's demand in period **`i`** (**`i`** =1, 2). These variables can be thought of as nodes in a network with the following data representing the COST and CAPACITY of the arcs connecting them:

```
FROM       TO          COST    CAPACITY

PLANT_1    CUST_1       10        75
PLANT_1    DEPOT1_1      7        75
PLANT_1    DEPOT2_1      8        75
DEPOT1_1   CUST_1        3        20
DEPOT2_1   CUST_1        2        10
PLANT_1    PLANT_2       2       100
DEPOT1_1   DEPOT1_2      1       100
DEPOT2_1   DEPOT2_2      1       100
PLANT_2    CUST_2       10        75
PLANT_2    DEPOT1_2      7        75
PLANT_2    DEPOT2_2      8        75
DEPOT1_2   CUST_2        3        20
DEPOT2_2   CUST_2        2        10
CUST_1     .             .         .
CUST_2     .             .         .
```

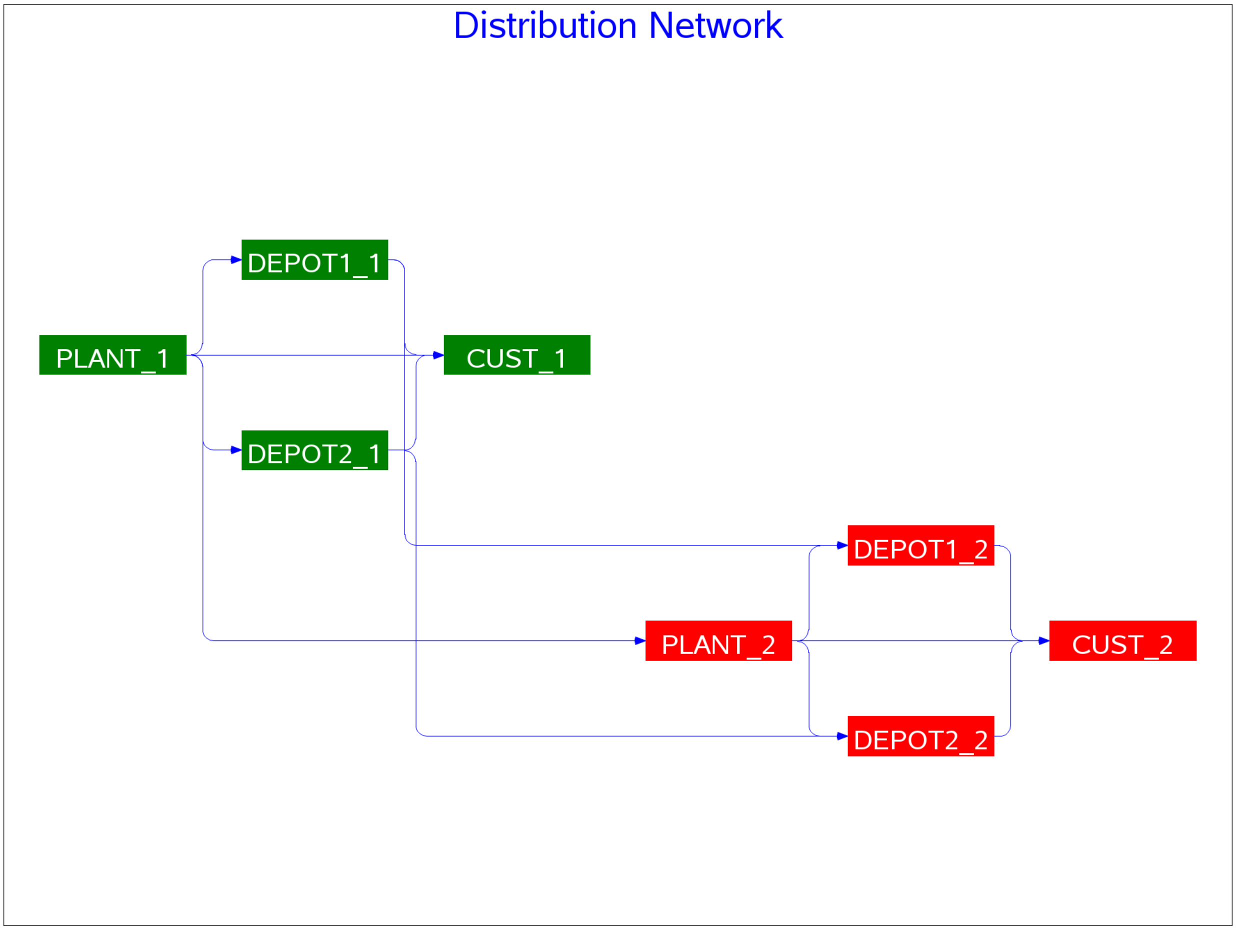
Suppose that you want to use PROC NETDRAW to draw the network corresponding to the preceding network flow problem and suppose also that you require the nodes to be placed in specific positions. The following program saves the network information along with the required node coordinates in the Network data set ARCS and invokes PROC NETDRAW to draw the network diagram shown in Output 9.14.1. The Network data set also contains a variable named _pattern, which specifies that pattern statement 1 be used for nodes relating to period 1 and pattern statement 2 be used for those relating to period 2.

```
data arcs;
   input from $  to $  _x_ _y_ _pattern;
   datalines;
PLANT_1  CUST_1      1  5  1
PLANT_1  DEPOT1_1    1  5  1
PLANT_1  DEPOT2_1    1  5  1
DEPOT1_1 CUST_1      2  6  1
DEPOT2_1 CUST_1      2  4  1
PLANT_1  PLANT_2     1  5  1
DEPOT1_1 DEPOT1_2    2  6  1
DEPOT2_1 DEPOT2_2    2  4  1
PLANT_2  CUST_2      4  2  2
PLANT_2  DEPOT1_2    4  2  2
PLANT_2  DEPOT2_2    4  2  2
DEPOT1_2 CUST_2      5  3  2
DEPOT2_2 CUST_2      5  1  2
CUST_1     .         3  5  1
CUST_2     .         6  2  2
;

title c=blue 'Distribution Network';
pattern1 v=s  c=green;
pattern2 v=s  c=red;
proc netdraw data=arcs graphics out=netout;
   actnet / act=from succ=to separatearcs
            ybetween = 4
            centerid
            ctext = white
```

```
            carcs=blue
            htext=2
            pcompress;
run;
```

NOTE: This network diagram can also be drawn by using suitably defined ZONE and ALIGN variables.

Output 9.14.1 Distribution Network

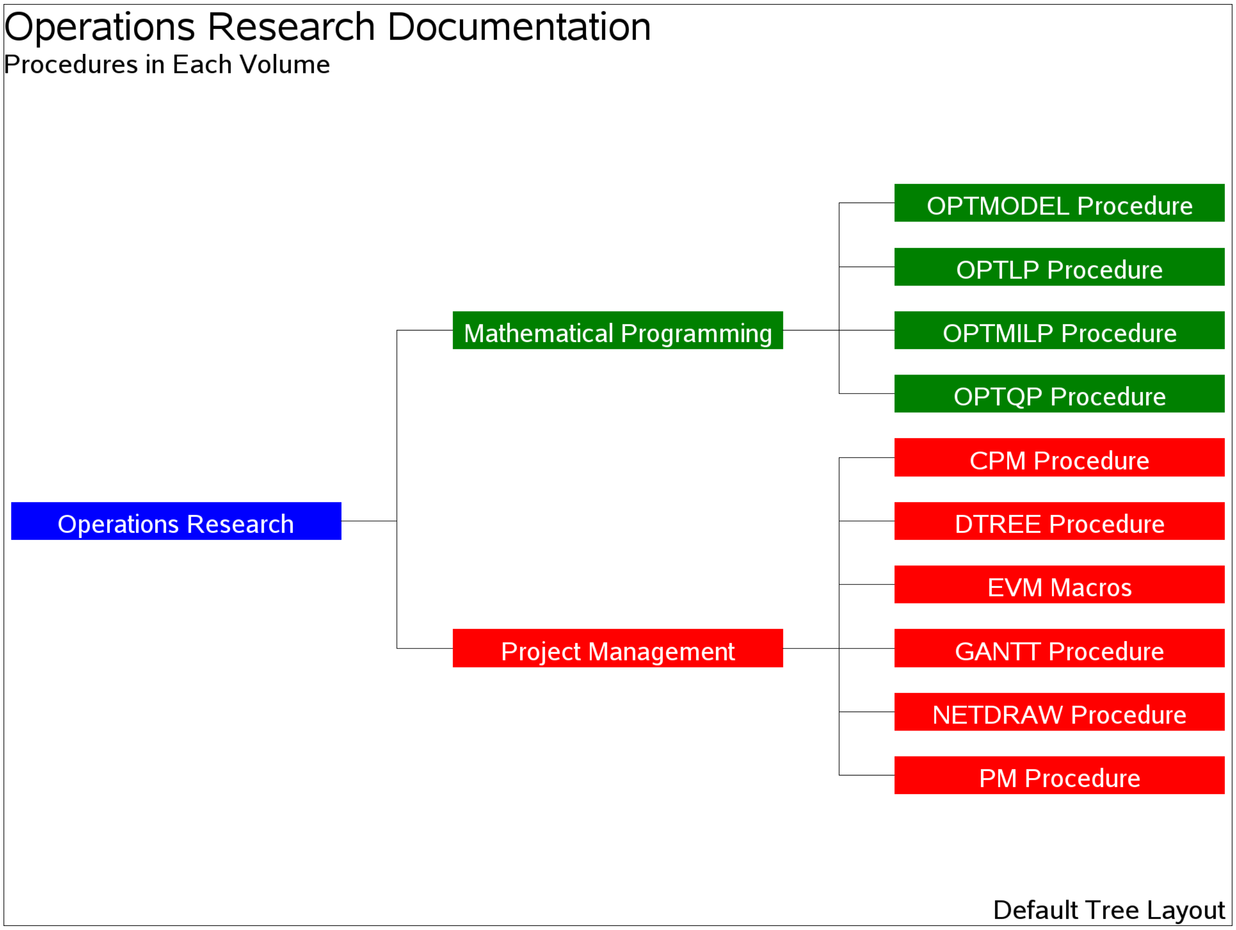


Example 9.15: Organizational Charts with PROC NETDRAW

This example illustrates using the TREE option to draw organizational charts. The Network data set, DOCUMENT, describes how the procedures are distributed between two volumes of the SAS/OR documentation. The structure can be visualized easily in a tree diagram. The data set DOCUMENT contains the parent-child relationship for each node of the diagram. For each node, a detailed description is contained in the variable ID. In addition, the variable _pattern specifies the pattern to be used for each node. PROC NETDRAW is invoked with the TREE option, which illustrates the organization of the documentation in the form of a tree diagram drawn from left to right. The CENTERID option centers text within each node. Arrowheads are not necessary for this diagram and are suppressed by specifying ARROWHEAD=0. Output 9.15.1 shows the resulting diagram.

```
data document;
   format parent child $8. id $24.;
   input parent $ child $ id & _pattern;
   datalines;
OR       MPBOOK     Operations Research       1
OR       PMBOOK     Operations Research       1
PMBOOK   CPM        Project Management        2
PMBOOK   DTREE      Project Management        2
PMBOOK   EVM        Project Management        2
PMBOOK   GANTT      Project Management        2
PMBOOK   NETDRAW    Project Management        2
PMBOOK   PM         Project Management        2
MPBOOK   OPTMODEL   Mathematical Programming  3
MPBOOK   OPTLP      Mathematical Programming  3
MPBOOK   OPTMILP    Mathematical Programming  3
MPBOOK   OPTQP      Mathematical Programming  3
CPM      .          CPM Procedure             2
DTREE    .          DTREE Procedure           2
EVM      .          EVM Macros                2
GANTT    .          GANTT Procedure           2
NETDRAW  .          NETDRAW Procedure         2
PM       .          PM Procedure              2
OPTMODEL .          OPTMODEL Procedure        3
OPTLP    .          OPTLP Procedure           3
OPTMILP  .          OPTMILP Procedure         3
OPTQP    .          OPTQP Procedure           3
;

pattern1 v=s c=blue;
pattern2 v=s c=red;
pattern3 v=s c=green;

title j=l h=3 'Operations Research Documentation';
title2 j=l h=2 'Procedures in Each Volume';
footnote j=r h=2 'Default Tree Layout ';
proc netdraw graphics data=document;
   actnet / act=parent
            succ=child
            id=(id)
            nodefid
            nolabel
            pcompress
            centerid
            tree
            xbetween=15
            ybetween=3
            arrowhead=0
            rectilinear
            carcs=black
            ctext=white
            htext=3;
   run;
```

Output 9.15.1 Organization of Documentation: Default TREE Layout

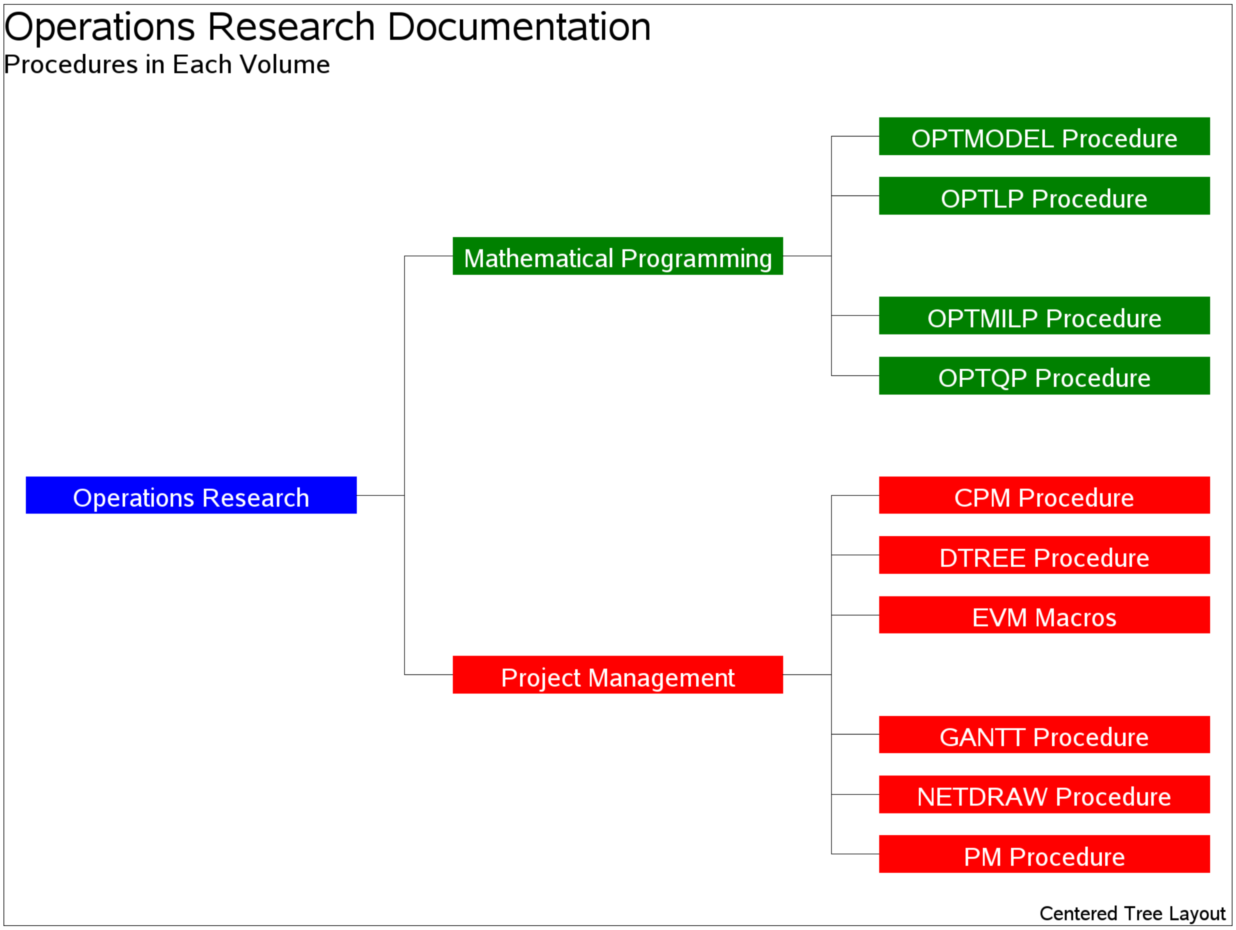


The procedure draws the tree compactly with the successors of each node being placed to the immediate right of the node, ordered from top to bottom in the order of occurrence in the Network data set. The next invocation of PROC NETDRAW illustrates the effect of the SEPARATESONS and CENTERSUBTREE options on the layout of the tree (see Output 9.15.2).

```
footnote j=r h=1.5 'Centered Tree Layout ';
proc netdraw graphics data=document;
   actnet / act=parent
            succ=child
            id=(id)
            nodefid
            nolabel
            pcompress
            novcenter
            centerid
            tree
            separatesons
            centersubtree
            xbetween=15
            ybetween=3
            arrowhead=0
```

```
            rectilinear
            carcs=black
            ctext=white
            htext=3.5;
   run;
```

Output 9.15.2 Organization of Documentation: Controlled TREE Layout



Example 9.16: Annotate Facility with PROC NETDRAW

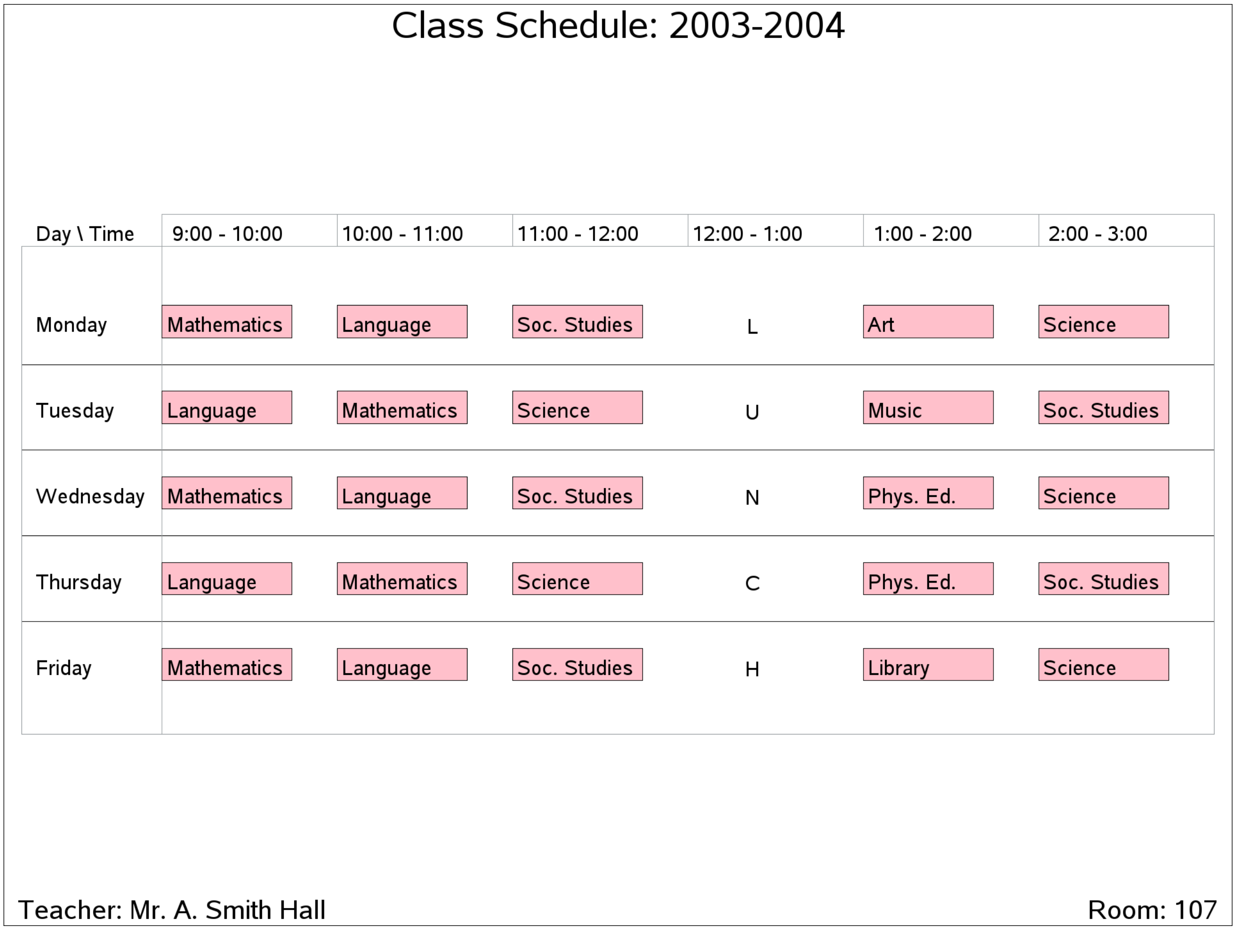
This example demonstrates the use of PROC NETDRAW for a nonstandard application. The procedure is used to draw a time table for a class of students. The days of the week are treated as different zones, and the times within a day are treated as different values of an alignment variable. The following DATA step defines a total of twenty activities, 'm1', ..., 'f5', which refer to the five different periods for the five different days of the week. The variable class contains the name of the subject taught in the corresponding period and day. Note that the periods are taught during the hours 1, 2, 3, 5, and 6; the fourth hour is set aside for lunch. The time axis is labeled with the format CLASSTIM, which is defined using PROC FORMAT. The USEFORMAT option in the ACTNET statement instructs PROC NETDRAW to use the explicit format specified for the time variable rather than the default format.

This example also illustrates the use of the Annotate facility with PROC NETDRAW. The data set ANNO labels the fourth period 'LUNCH.' The positions for the text are specified using data coordinates that refer to the (_X_, _Y_) grid used by PROC NETDRAW. Thus, for example X = '4' identifies the *x* coordinate for the annotated text to be the fourth period, and the *y* coordinates are set appropriately. The resulting time table is shown in Output 9.16.1.

```
 /* Define format for the ALIGN= variable */
proc format;
   value classtim 1 = ' 9:00 - 10:00'
                  2 = '10:00 - 11:00'
                  3 = '11:00 - 12:00'
                  4 = '12:00 - 1:00 '
                  5 = ' 1:00 - 2:00 '
                  6 = ' 2:00 - 3:00 ';
   run;

data schedule;
   format day $9. class $12. ;
   input day $ class &  time daytime $ msucc $;
   format time classtim.;
   label day = "Day \ Time";
   datalines;
Monday     Mathematics    1   m1   .
Monday     Language       2   m2   .
Monday     Soc. Studies   3   m3   .
Monday     Art            5   m4   .
Monday     Science        6   m5   .
Tuesday    Language       1   t1   .
Tuesday    Mathematics    2   t2   .
Tuesday    Science        3   t3   .
Tuesday    Music          5   t4   .
Tuesday    Soc. Studies   6   t5   .
Wednesday  Mathematics    1   w1   .
Wednesday  Language       2   w2   .
Wednesday  Soc. Studies   3   w3   .
Wednesday  Phys. Ed.      5   w4   .
Wednesday  Science        6   w5   .
Thursday   Language       1   th1  .
Thursday   Mathematics    2   th2  .
Thursday   Science        3   th3  .
Thursday   Phys. Ed.      5   th4  .
Thursday   Soc. Studies   6   th5  .
Friday     Mathematics    1   f1   .
Friday     Language       2   f2   .
Friday     Soc. Studies   3   f3   .
Friday     Library        5   f4   .
Friday     Science        6   f5   .
;

data anno;
   /* Set up required variable lengths, etc. */
   length function color style   $8;
   length xsys ysys hsys         $1;
   length when position          $1;

   xsys     = '2';
```

```
      ysys     = '2';
      hsys     = '4';
      when     = 'a';

      function = 'label   ';
      x = 4;
      size = 2;
      position = '5';
      y=5; TEXT='L'; output;
      y=4; TEXT='U'; output;
      y=3; TEXT='N'; output;
      y=2; TEXT='C'; output;
      y=1; TEXT='H'; output;
      run;

   pattern1 v=s c=pink;
   title 'Class Schedule: 2003-2004';
   footnote j=l h=2 '  Teacher: Mr. A. Smith Hall'
            j=r h=2 'Room: 107  ';
   proc netdraw graphics data=schedule anno=anno;
      actnet / act=daytime
               succ=msucc
               id=(class)
               nodefid nolabel
               zone=day
               align=time
               useformat
               linear
               pcompress
               coutline=black
               hmargin = 2 vmargin = 2
               htext=2;
      run;
```

Output 9.16.1 Use of the Annotate Facility



Example 9.17: AOA Network Using the Annotate Facility

This example illustrates the use of the Annotate Facility to draw an Activity-on-Arc network. First, PROC NETDRAW is invoked with explicit node positions for the vertices of the network. The ALIGN= and ZONE= options are used to provide horizontal and vertical axes as a frame of reference. The resulting diagram is shown in Output 9.17.1.

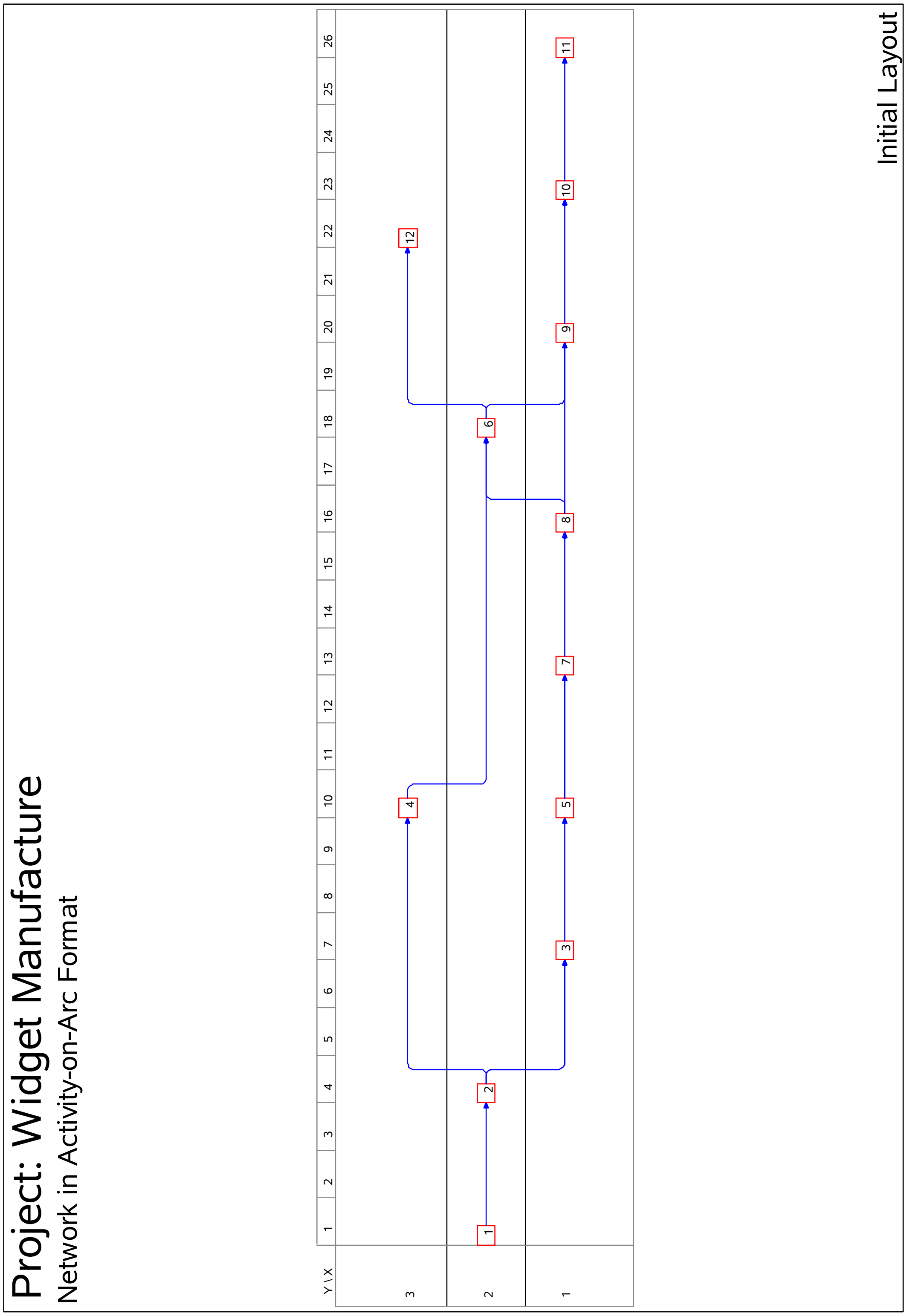
```
data widgaoa;
   format task $12. ;
   input task & days tail head _x_ _y_;
   datalines;
Approve Plan    5   1   2   1  2
Drawings       10   2   3   4  2
Study Market    5   2   4   4  2
Write Specs     5   2   3   4  2
Prototype      15   3   5   7  1
Mkt. Strat.    10   4   6  10  3
Materials      10   5   7  10  1
Facility       10   5   7  10  1
Init. Prod.    10   7   8  13  1
```

```
Evaluate      10   8   9  16  1
Test Market   15   6   9  18  2
Changes        5   9  10  20  1
Production     0  10  11  23  1
Marketing      0   6  12  19  2
Dummy          0   8   6  16  1
.              .  11   .  26  1
.              .  12   .  22  3
;

pattern1 v=e c=red;
title j=l h=3 ' Project: Widget Manufacture';
title2 j=l h=2 ' Network in Activity-on-Arc Format';
footnote j=r h=2 'Initial Layout ';

proc netdraw graphics data=widgaoa out=netout;
   actnet / act=tail
            succ=head
            id=(tail)
            align=_x_
            zone=_y_
            ybetween = 10
            nodefid
            nolabel
            pcompress
            htext=2;
   label _y_=' Y \ X ';
run;
```

In Output 9.17.1, the arc leading from vertex '4' to vertex '6' has two turning points: (10.5, 3) and (10.5, 2). Suppose that you want the arc to be routed differently, to provide a more symmetric diagram. The next DATA step creates a data set, NETIN, which changes the *x* coordinates of the turning points to 16.5 instead of 10.5. Further, two Annotate data sets are created: the first one labels the nodes outside the boxes, either to the top or to the bottom, and the second one sets labels for the arcs. PROC NETDRAW is then invoked with the combined Annotate data set to produce the diagram shown in Output 9.17.2.

```
data netin;
  set netout;
  if _from_=4 and _to_=6 and _seq_>0 then _x_=16.5;
  run;

data anno1;
   set netout;
   if _seq_=0;
   /* Set up required variable lengths, etc. */
   length function color style   $8;
   length xsys ysys hsys         $1;
   length when position          $1;
   length TEXT                   $12;
   xsys     = '2';
   ysys     = '2';
   hsys     = '4';
   when     = 'a';
   function = 'label   ';
   size = 2;
```

Output 9.17.1 Activity-on-Arc Format



```
   position = '5';
   TEXT = left(put(tail, f2.));
   x=_x_;
   if _y_ = 1 then y=_y_-.3;
   else            y=_y_+.5;
   run;

data anno2;
   /* Set up required variable lengths, etc. */
   length function color style   $8;
   length xsys ysys hsys         $1;
   length when position          $1;
   length TEXT                   $12;
   xsys     = '2';
   ysys     = '2';
   hsys     = '4';
   when     = 'a';
   function = 'label   ';
   size = 2;
   position = '5';
   x=2.5;  y=1.8;  TEXT='Approve Plan'; output;
   x=5.5;  y=.8;   TEXT='Drawings';     output;
   x=5.7;  y=1.4;  TEXT='Write Specs';  output;
   x=7;    y=3.4;  TEXT='Study Market';  output;
   x=8.5;  y=.8;   TEXT='Prototype';    output;
   x=11.5; y=1.4;  TEXT='Facility';     output;
   x=11.5; y=.8;   TEXT='Materials';    output;
   x=14.5; y=.9;   TEXT='Init. Prod';   output;
   x=13.5; y=3.4;  TEXT='Mkt. Strat.';  output;
   x=18;   y=.8;   TEXT='Evaluate';     output;
   x=21.5; y=.8;   TEXT='Changes';      output;
   x=24.5; y=.8;   TEXT='Production';   output;
   x=20;   y=3.4;  TEXT='Marketing';    output;
   position=6;
   x=16.6; y=1.5;  TEXT='Dummy';        output;
   x=18.6; y=1.5;  TEXT='Test Market';  output;
   ;

data anno;
   set anno1 anno2;
   run;

footnote j=r h=2 'Annotated and Modified Layout ';
pattern1 v=s c=red;

proc netdraw graphics data=netin anno=anno;
   actnet / nodefid
            nolabel
            boxwidth=1
            pcompress
            novcenter
            vmargin=20
            xbetween=10;
run;
```

Example 9.18: Branch and Bound Trees

This example illustrates a nonstandard use of PROC NETDRAW. The TREE option in PROC NETDRAW is used to draw a branch and bound tree such as one that you obtain in the solution of an integer programming problem. Refer to Chapter 4, "The LP Procedure" (*SAS/OR User's Guide: Mathematical Programming Legacy Procedures*) for a detailed discussion of branch and bound trees. The data used in this example were obtained from one particular invocation of PROC LP.

The data set NET (created in the following DATA step) contains information pertaining to the branch and bound tree. Each observation of this data set represents a particular iteration of the integer program, which can be drawn as a node in the tree. The variable node names the problem. The variable object gives the objective value for that problem. The variable problem identifies the *parent* problem corresponding to each node; for example, since the second and the seventh observations have problem equal to '-1' and '1', respectively, it indicates that the second and the seventh problems are derived from the first iteration. Finally, the variable _pattern specifies the pattern of the nodes based on the status of the problem represented by the node.

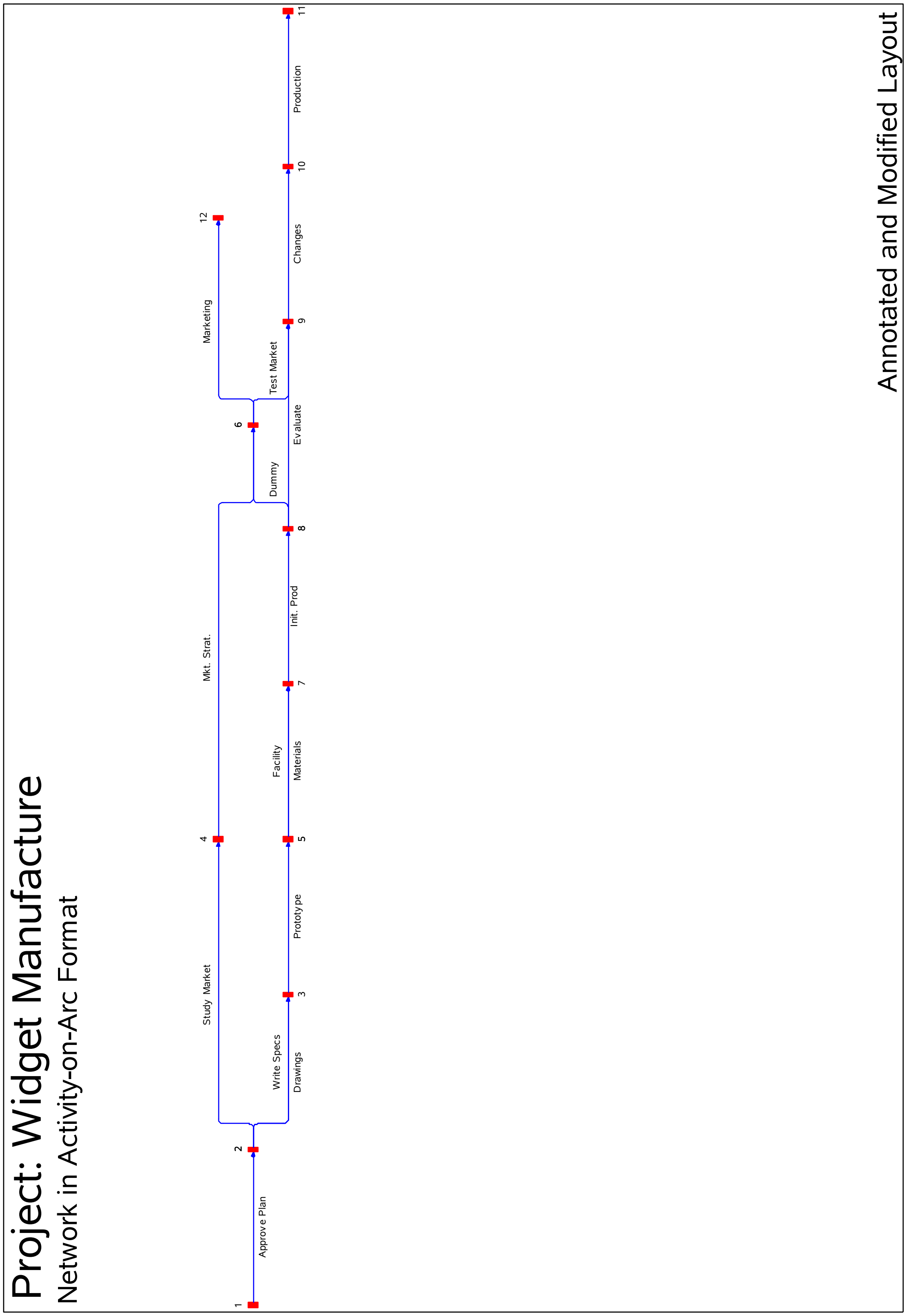
```
data net;
   input node problem cond $10. object;
   if cond="ACTIVE"           then _pattern=1;
   else if cond="SUBOPTIMAL" then _pattern=2;
   else                           _pattern=3;
datalines;
 1       0     ACTIVE         4
 2      -1     ACTIVE         4
 3       2     ACTIVE         4
 4      -3     ACTIVE 4.3333333
 5       4 SUBOPTIMAL         5
 6       3   FATHOMED 4.3333333
 7       1     ACTIVE         4
 8      -7     ACTIVE         4
 9      -8   FATHOMED 4.3333333
10       8   FATHOMED 4.3333333
11       7     ACTIVE         4
12     -11   FATHOMED 4.3333333
13      11   FATHOMED       4.5
;
```

The next DATA step (which creates the data set LOGIC) uses this child-parent information to format the precedence relationships as expected by PROC NETDRAW. Next, the two data sets are merged together to create the Network input data set (BBTREE) for PROC NETDRAW. The ID variable in the data set BBTREE is formatted to contain both the iteration number and the objective value.

Finally, PROC NETDRAW is invoked with the TREE, ROTATE, and ROTATETEXT options to produce a branch and bound tree shown in Output 9.18.1. Note that the ROTATE and ROTATETEXT options produce a rotated graph with a top-down orientation.

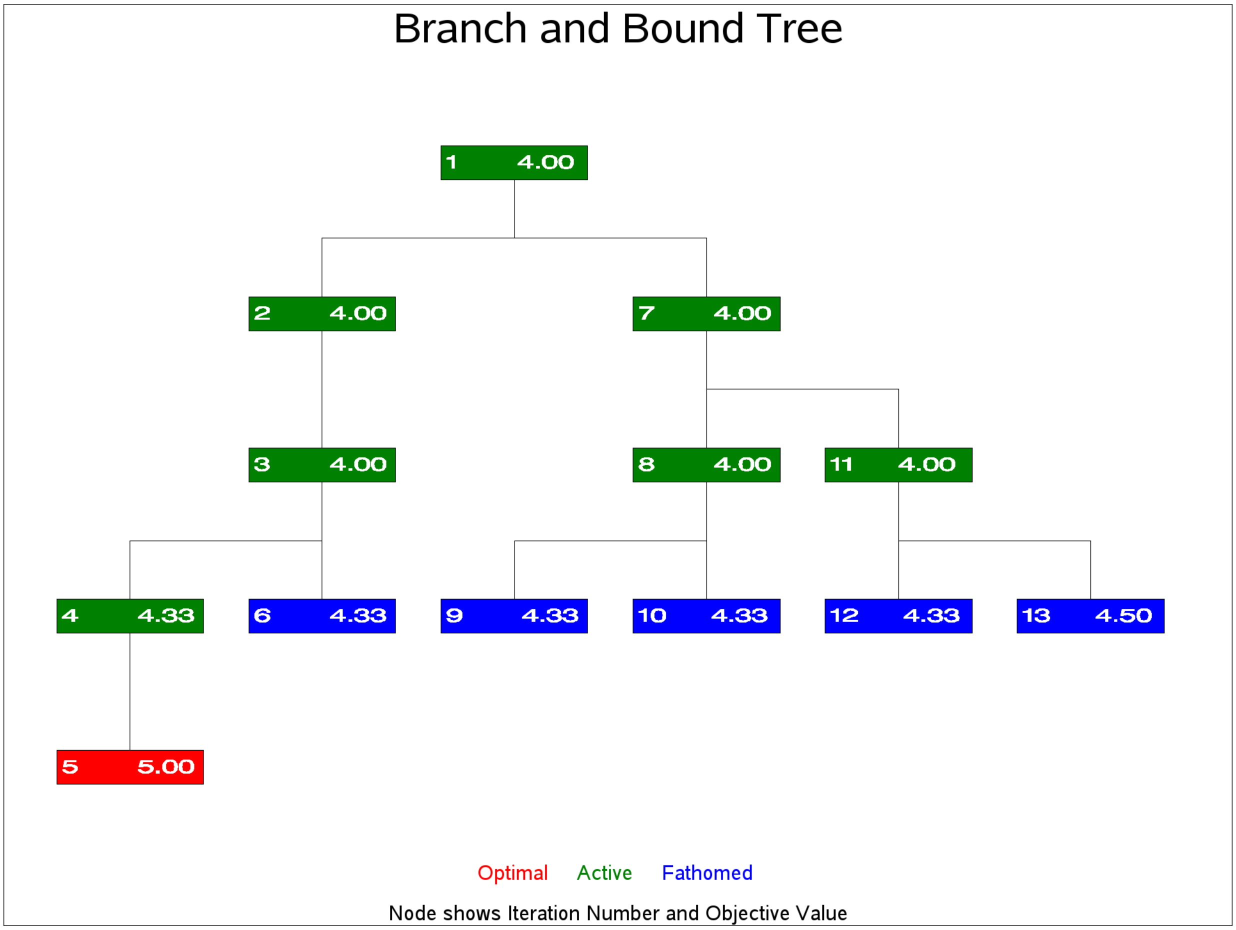
```
/* set precedence relationships
   using child-parent information */
data logic;
   keep node succ;
```

Output 9.17.2 Activity-on-Arc Format: Annotated Diagram



```
   set net(firstobs=2);
   succ=node;
   node=abs(problem);
   run;

proc sort data=logic;
   by node;
   run;

/* combine the logic data and the node data */
/* set ID values                              */
data bbtree;
  length id $ 9;
  merge logic net; by node;
  if node < 10 then id=put(node,1.)||put(object,f8.2);
  else              id=put(node,2.)||put(object,f7.2);
  run;

goptions border rotate=portrait;
pattern1 v=s c=green;
pattern2 v=s c=red;
pattern3 v=s c=blue;

title h=3    j=c 'Branch and Bound Tree';
title2    ' ';
footnote1 h=1.5 j=c c=red   'Optimal  '
                    c=green '   Active  '
                    c=blue  '   Fathomed ';
footnote2 ' ';
footnote3 h=1.5 ' Node shows Iteration Number and Objective Value ';
proc netdraw data=bbtree graphics out=bbout;
   actnet /activity=node
           successor=succ
           id=(id)
           nodefid
           nolabel
           ctext=white
           coutline=black
           carcs=black
           xbetween=15
           ybetween=3
           compress
           font=swiss
           rectilinear
           tree
           rotate
           rotatetext
           arrowhead=0
           htext=2;
   run;
```

Output 9.18.1 Branch and Bound Tree

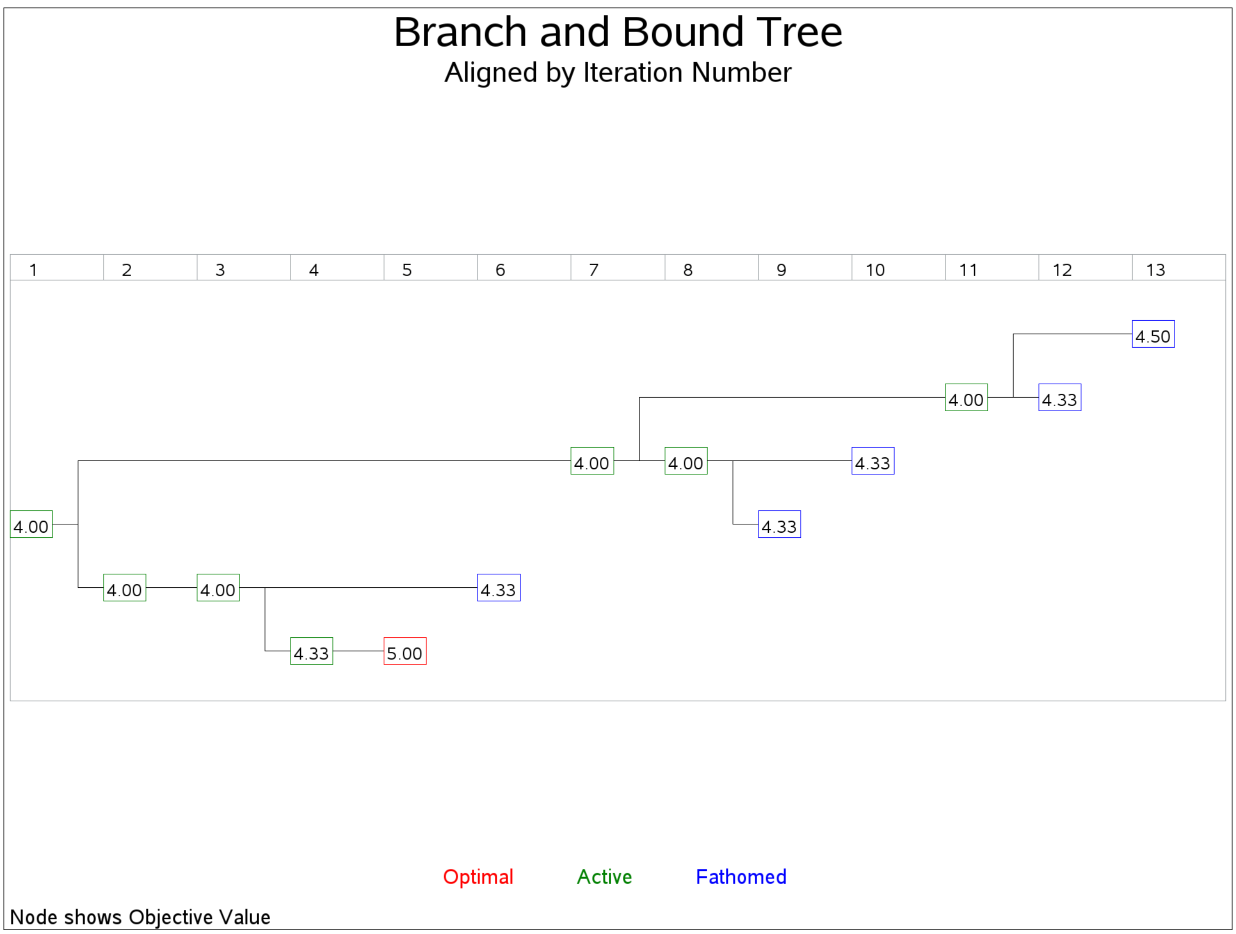


In the next invocation, PROC NETDRAW uses a modified layout of the nodes to produce a diagram where the nodes are aligned according to the iteration number. The following program uses the Layout data set produced in the previous invocation of PROC NETDRAW. The same *y* coordinates are used; but the *x* coordinates are changed to equal the iteration number. Further, the ALIGN= specification produces a time axis that labels each level of the diagram with the iteration number. Each node is labeled with the objective value. The resulting diagram is shown in Output 9.18.2.

```
data netin;
   set bbout;
   if _seq_ = 0; drop _seq_ ;
   _x_ = _from_;
   id = substr(id, 3);
   run;

goptions rotate=landscape;
title h=3   'Branch and Bound Tree';
title2 h=2  'Aligned by Iteration Number';
footnote1 h=1.5 j=c  c=red   'Optimal      '
                     c=green '      Active      '
                     c=blue  '      Fathomed ';
footnote2 ' ';
```

```
footnote3 j=l h=1.5 ' Node shows Objective Value ';
pattern1 v=e c=green;
pattern2 v=e c=red;
pattern3 v=e c=blue;
proc netdraw data=netin graphics;
   actnet /id=(id)
           ctext=black
           carcs=black
           align = _from_
           frame
           pcompress
           rectilinear
           arrowhead=0
           nodefid
           nolabel
           htext=2.5
           xbetween=8;
   run;
```

Output 9.18.2 Branch and Bound Tree: Aligned by Iteration Number



Statement and Option Cross-Reference Tables

The next two tables reference the options in the NETDRAW procedure that are illustrated by the examples in this section. Note that all the options are specified on the ACTNET statement.

Table 9.9 Options Specified in Examples 5.1–5.9

Option	1	2	3	4	5	6	7	8	9
ACTIVITY=	X	X	X	X	X	X	X	X	X
ALIGN=									X
ARROWHEAD=						X			
AUTOREF									X
BOXHT=						X	X		
BOXWIDTH=					X				X
CARCS=		X	X	X				X	X
CCRITARCS=			X	X				X	X
CCRITOUT=			X					X	
CENTERID						X			
COMPRESS		X		X					
COUTLINE=			X					X	
CREF=									X
CTEXT=			X						
DATA=	X	X	X	X	X	X	X	X	X
DP							X		
DURATION=	X		X	X			X		
FONT=				X					
FRAME									X
GRAPHICS		X	X	X	X	X	X	X	X
HMARGIN=									X
HPAGES=					X				X
HTEXT=		X			X	X	X	X	X
HTRACKS=							X		
ID=					X			X	X
LAG=						X			
LINEPRINTER	X								
LREF=									X
LWCRIT=			X						
LWIDTH=		X	X						
NODEFID					X			X	X
NOLABEL					X			X	X
NOPAGENUMBER					X				X
NOVCENTER				X					X
PATTERN=								X	
PCOMPRESS				X		X	X	X	X
SEPARATEARCS			X	X	X	X	X	X	X
SHOWBREAK									X
SHOWSTATUS								X	X

Table 9.9 (continued)

Option	1	2	3	4	5	6	7	8	9
SUCCESSOR=	X	X	X	X	X	X	X	X	X
TIMESCALE									X
VMARGIN=									X
VPAGES=					X				X
XBETWEEN=						X	X		
YBETWEEN=					X	X		X	X

Table 9.10 Options Specified in Examples 5.10–5.18

Option	10	11	12	13	14	15	16	17	18
ACTIVITY=	X	X	X	X	X	X	X	X	X
ALIGN=							X	X	X
ANNOTATE=							X	X	
ARROWHEAD=						X			X
AUTOREF		X							
BOXHT=				X					
BOXWIDTH=	X	X						X	
BREAKCYCLE			X						
CARCS=	X	X	X	X	X	X			X
CAXIS=		X							
CCRITARCS=		X							
CENTERID			X	X	X	X			
CENTERSUBTREE						X			
COMPRESS	X								X
COUTLINE=			X	X			X		X
CREF=		X							
CREFBRK=	X								
CTEXT=			X	X	X	X			X
DATA=	X	X	X	X	X	X	X	X	X
DP	X								
DURATION=	X								
FONT=	X								X
FRAME	X								X
GRAPHICS	X	X	X	X	X	X	X	X	X
HMARGIN=			X				X		
HTEXT=	X	X	X	X	X	X	X	X	X
ID=	X	X	X	X		X	X	X	X
LINEAR	X	X					X		
LREF=		X							
LREFBRK=	X								
MININTERVAL=		X							
NOARROWFILL			X						
NODEFID	X	X	X	X		X	X	X	X
NOLABEL	X	X	X	X		X	X	X	X

Table 9.10 (continued)

Option	10	11	12	13	14	15	16	17	18
NOVCENTER	X		X			X		X	
OUT=				X	X			X	X
PATTERN=			X						
PCOMPRESS	X	X	X	X	X	X	X	X	X
RECTILINEAR			X			X			X
REFBREAK	X								
ROTATE									X
ROTATETEXT									X
SEPARATEARCS	X	X		X	X				
SEPARATESONS						X			
SUCCESSOR=	X	X	X	X	X	X	X	X	X
TIMESCALE	X	X							
TREE						X			X
USEFORMAT							X		
VMARGIN=	X		X				X	X	
XBETWEEN=			X			X		X	X
YBETWEEN=			X	X	X	X		X	X
ZONE=		X					X	X	
ZONEPAT		X							
ZONESPACE		X							

References

Even, S. (1979). *Graph Algorithms*. Rockville, MD: Computer Science Press.

Kuhfeld, W. F. (2010). *Statistical Graphics in SAS: An Introduction to the Graph Template Language and the Statistical Graphics Procedures*. Cary, NC: SAS Institute Inc.

Chapter 10
The Earned Value Management Macros

Contents

Overview: Earned Value Management Macros

Earned value refers to the amount of work that has been completed within a project to date. In Earned Value Management (EVM), this earned value is compared to the work that was planned, in order to measure project performance and to estimate future costs and project completion.

To see why an earned value analysis might be of interest, consider a four-week project with a planned completion cost of $10,000. After one week, the Planned Value (PV) is one quarter of $10,000, or $2,500. Assume that only $2,000 has been spent so far. This is the Actual Cost (AC). Also assume that only one fifth of the project has been completed. This gives an Earned Value (EV) of $2,000, which is one fifth of the original planned completion cost. The amount of work accomplished for the cost incurred is as expected, but the project is progressing more slowly than expected. In other words, the project is behind schedule, but within its budgetary constraints.

Based upon the performance of the project so far, EVA can be used to forecast the funds required to complete the project, and to predict the project completion date. Historically, earned value measurements have been found to be reliable indicators of future project performance as early as the 15 to 20 % completion point (see Fleming and Koppelman 2000).

The cost performance is good for the four-week project, so the project is still expected to cost $10,000 on completion. However, based on the schedule performance, the project duration is predicted to be five weeks instead of four weeks as originally planned. Additional estimates can provide upper and lower bounds on the project costs; these estimates are discussed in the section "Getting Started: Earned Value Management Macros" on page 781.

In this chapter two distinct sets of macros are described: one for analysis and one for reporting. The first set, consisting of the %EVA_PLANNED_VALUE, %EVA_EARNED_VALUE, %EVA_METRICS, and %EVA_TASK_METRICS macros, is used to analyze schedule and cost data and to derive earned value metrics. The second set, consisting of the %EVG_COST_PLOT, %EVG_SCHEDULE_PLOT, %EVG_INDEX_PLOT, %EVG_VARIANCE_PLOT, %EVG_GANTT_CHART and %EVG_WBS_CHART macros, is used to graphically represent the results from the first set.

NOTE: To use the graphical macros, SAS/GRAPH must be licensed.

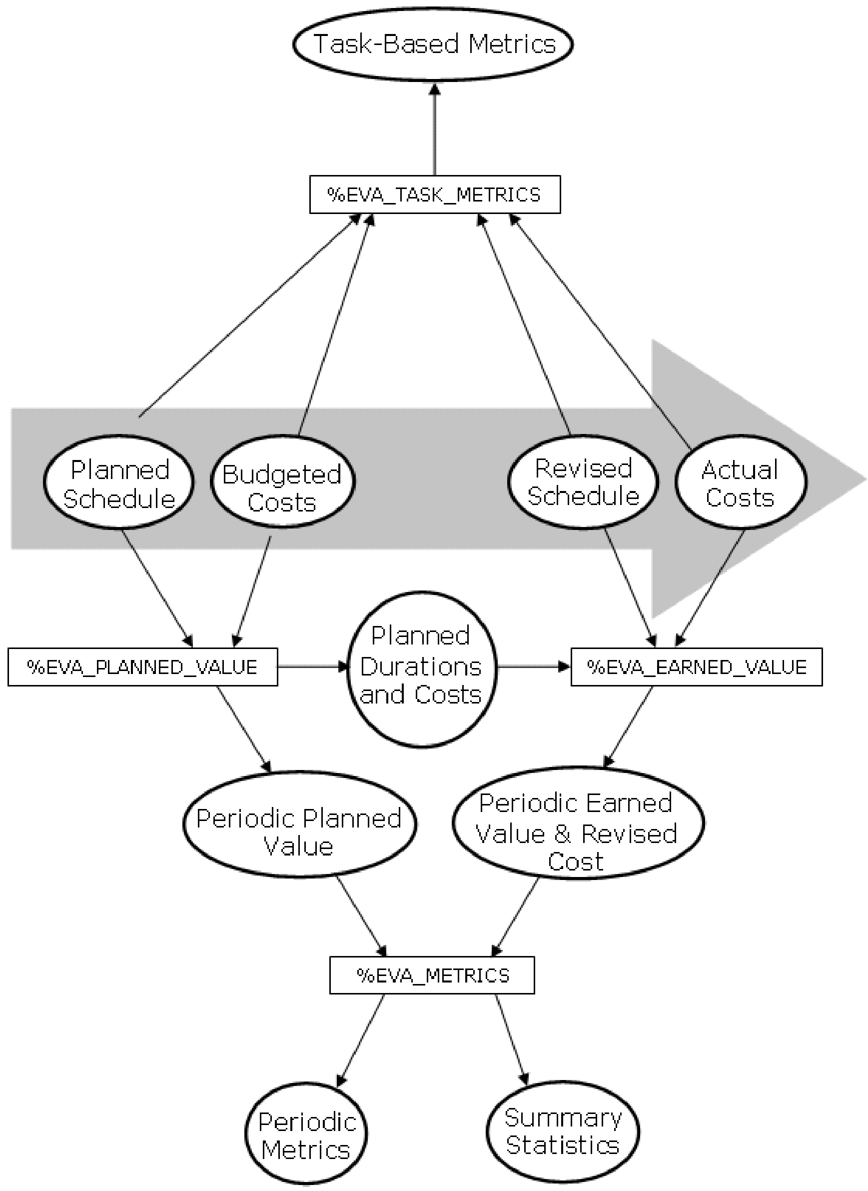
The %EVA_PLANNED_VALUE macro computes the periodic planned value for the project. The periodic planned value is the periodic cost that is to be incurred by the project and is derived from the planned schedule and costs, and the workday, calendar, and holiday data sets. (These inputs are provided by the user.) The costs associated with an activity can be incurred continuously and/or at discrete time points. Continuously incurred costs are modeled by specifying a cost rate. For discretely incurred costs, the cost of the activity can be apportioned at the beginning (100-0), at the end (0-100), both equally (50-50), or both asymmetrically (custom), as desired. In general, costs can be incurred at evenly distributed time points over the duration of the associated activity, according to the number of weights specified. This macro can be run at any time —before, during, or after project execution.

Once the project is under way and actual schedule and cost information is available, the %EVA_EARNED_VALUE macro can be invoked to produce the revised periodic cost, along with periodic earned value. The set of periodic costs produced by %EVA_PLANNED_VALUE and %EVA_EARNED_VALUE can then be used in the %EVA_METRICS macro to generate earned value performance measurements and summary statistics for one or more status dates. In addition, the %EVA_METRICS macro computes cumulative costs, indices, and variances.

The %EVA_TASK_METRICS macro produces task-based performance measurements including Cost and Schedule Variance (CV and SV, respectively) and Cost and Schedule Performance Index (CPI and SPI, respectively).

The flow of data through the analysis macros is illustrated in Figure 10.1. The large arrow indicates the passage of time between the planned schedule and budget and the updated schedule and budget.

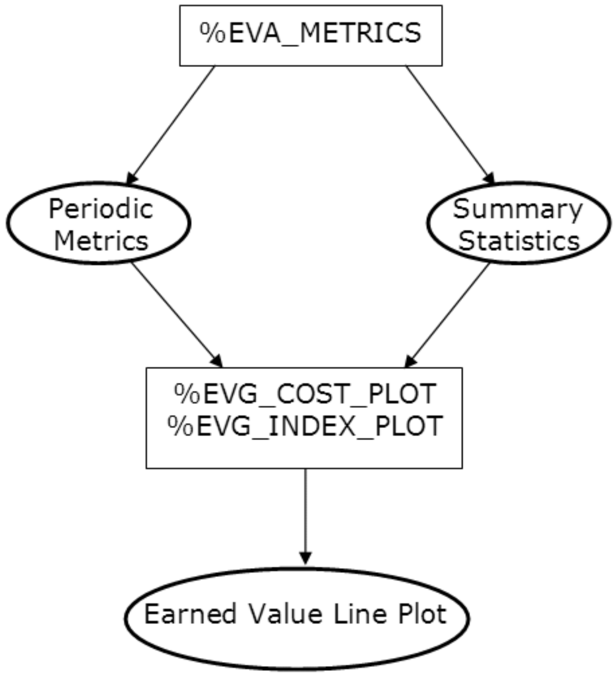
Figure 10.1 Data Flow for Analysis Macros



The first of the graphics macros are %EVG_COST_PLOT and %EVG_INDEX_PLOT. %EVG_COST_PLOT is used to produce line plots of planned value, earned value, actual cost, and revised cost over time, using the output from the %EVA_METRICS macro.

The %EVG_INDEX_PLOT macro displays line plots of the Cost and Schedule Performance Indices (CPI and SPI, respectively), also using the output from the %EVA_METRICS macro. The CPI is the earned value per actual cost and the SPI is the earned value per planned value. Figure 10.2 shows the data flow for the %EVG_INDEX_PLOT and the %EVG_COST_PLOT macros.

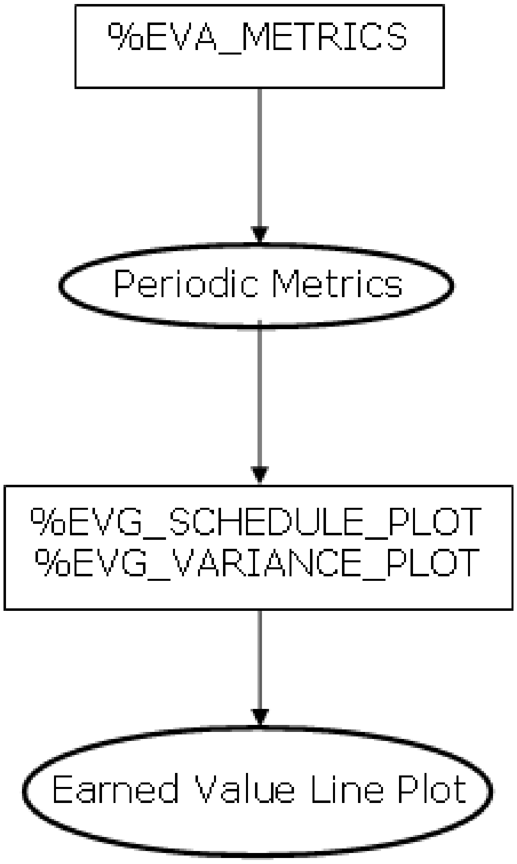
Figure 10.2 Data Flow for %EVG_COST_PLOT and %EVG_INDEX_PLOT Macros



The %EVG_SCHEDULE_PLOT macro is a schedule-oriented version of %EVG_COST_PLOT. The plot produced by %EVG_SCHEDULE_PLOT shows an estimate for the end date of the project. This estimate is an extrapolation of the current earned value to the BAC (Budget at Completion).

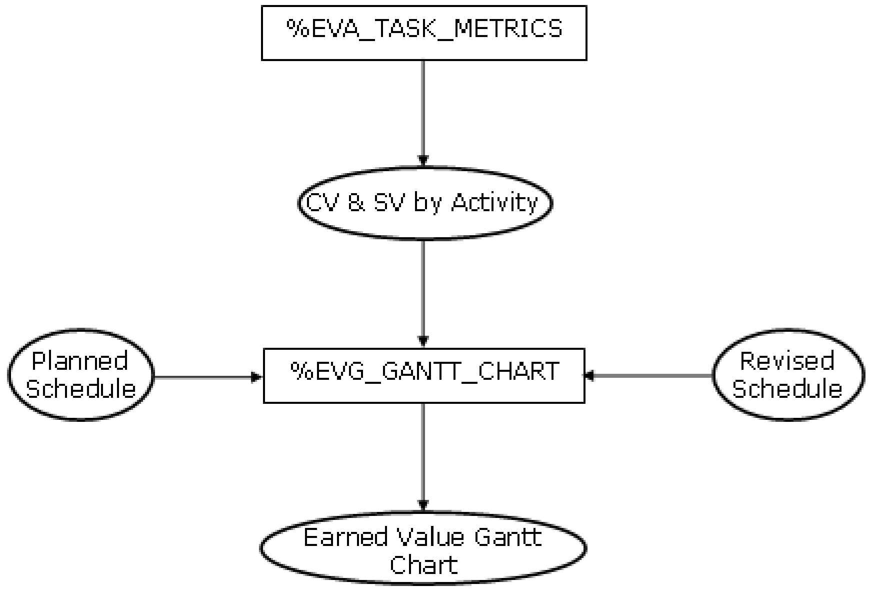
%EVG_VARIANCE_PLOT uses the output from %EVA_METRICS to produce line plots for the project Cost and Schedule Variance over time. The CV is computed as the difference between the earned value and actual cost, whereas the SV is the difference between earned value and planned value. Figure 10.3 shows the data flow for the %EVG_SCHEDULE_PLOT and %EVG_VARIANCE_PLOT macros.

Figure 10.3 Data Flow for %EVG_SCHEDULE_PLOT and %EVG_VARIANCE_PLOT Macros



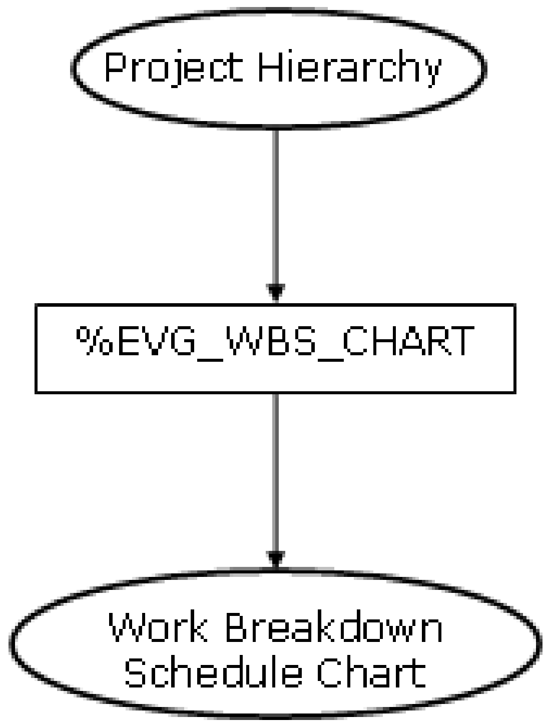
The %EVG_GANTT_CHART macro uses the planned and revised schedule and budget, along with the output from the %EVA_TASK_METRICS macro, to produce a Gantt chart of the project activities, along with the CV and SV for each activity. A pictorial description of the data flow is given in Figure 10.4.

Figure 10.4 Data Flow for %EVG_GANTT_CHART Macro



Finally, for multiprojects, or projects with a task/subtask hierarchy, the %EVG_WBS_CHART macro is used to display the Work Breakdown Structure. The input data set for this macro contains the activity, the project to which it belongs (or alternatively, the Work Breakdown Structure code), and any miscellaneous data for each activity that is to be carried into the graphical output. The data flow is shown in Figure 10.5.

Figure 10.5 Data Flow for %EVG_WBS_CHART Macro



Getting Started: Earned Value Management Macros

Analysis

Generating Periodic Data Sets

Consider the following set of activities that constitute a software multiproject. The Project variable defines the project hierarchy and the Succesr1 and Succesr2 variables define the precedence relationships. The complete project data is shown in Figure 10.6.

```
data softin;
   input Project  $10.
         Activity $11.
         Description $22.
         Duration
         Succesr1 $9.
         Succesr2 $9.;
   datalines;
          SWPROJ     Software project       .
DEBUG     RECODE     Recoding               5   DOCEDREV  QATEST
DOC       DOCEDREV   Doc. Edit and Revise  10   PROD
```

```
DOC       PRELDOC    Prel. Documentation   15  DOCEDREV  QATEST
MISC      MEETMKT    Meet Marketing         0  RECODE
MISC      PROD       Production             1
SWPROJ    DEBUG      Debug & Code Fixes     .
SWPROJ    DOC        Doc. Subproject        .
SWPROJ    MISC       Miscellaneous          .
SWPROJ    TEST       Test Subproject        .
TEST      QATEST     QA Test Approve       10  PROD
TEST      TESTING    Initial Testing       20  RECODE
;
```

Figure 10.6 Software Schedule Input SOFTIN

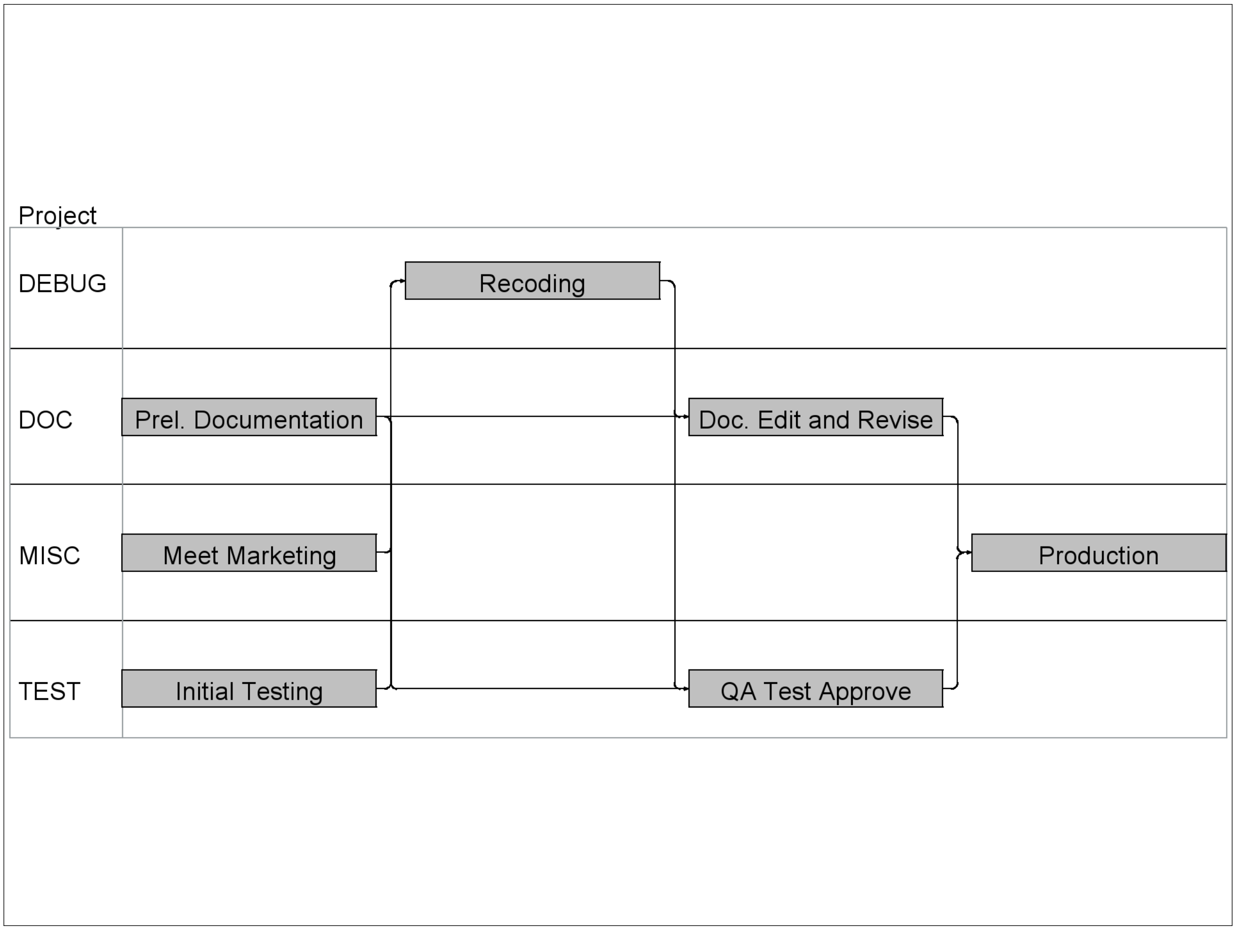
Software Schedule Input

Obs	Project	Activity	Description	Duration	Succesr1	Succesr2
1		SWPROJ	Software project	.		
2	DEBUG	RECODE	Recoding	5	DOCEDREV	QATEST
3	DOC	DOCEDREV	Doc. Edit and Revise	10	PROD	
4	DOC	PRELDOC	Prel. Documentation	15	DOCEDREV	QATEST
5	MISC	MEETMKT	Meet Marketing	0	RECODE	
6	MISC	PROD	Production	1		
7	SWPROJ	DEBUG	Debug & Code Fixes	.		
8	SWPROJ	DOC	Doc. Subproject	.		
9	SWPROJ	MISC	Miscellaneous	.		
10	SWPROJ	TEST	Test Subproject	.		
11	TEST	QATEST	QA Test Approve	10	PROD	
12	TEST	TESTING	Initial Testing	20	RECODE	

The project network diagram showing the precedence relationships can be generated with the following code, using the NETDRAW procedure (see Chapter 9 for additional information). The resulting diagram is shown in Figure 10.7. (Note that the first DATA step removes the root task to simplify the diagram.)

```
data softabridged;
   set softin;
   if activity ne 'SWPROJ' and project ne 'SWPROJ' then output;
run;


proc netdraw data=softabridged out=ndout graphics;
   actnet/act=activity
   id=(description)
   nodefid
   nolabel
   succ=(succesr1 succesr2)
   lwidth=2
   coutline=black
   ctext=black
   carcs=black
   cnodefill=ltgray
   zone=project
   zonespace
   htext=3 pcompress
   centerid;
run;
```

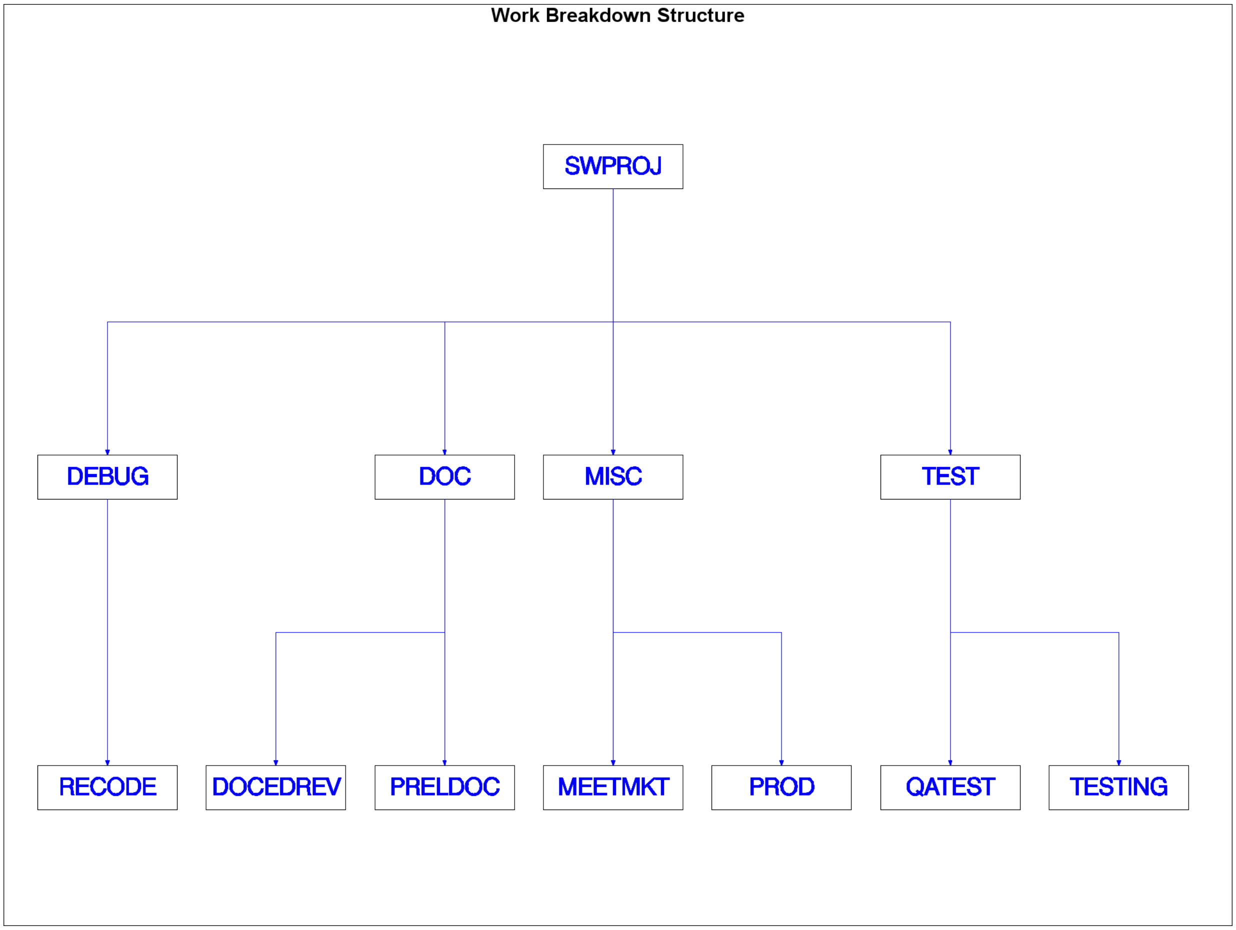
Figure 10.7 Project Network Diagram Using PROC NETDRAW



The project hierarchy, or Work Breakdown Structure (WBS), can be visualized with the following code. The graphical output is shown in Figure 10.8. The details for the use of the %EVG_WBS_CHART macro are given in the section "Syntax: Earned Value Management Macros" on page 809.

```
%evg_wbs_chart(
   structure=softin,
   activity=activity,
   project=project
);
```

Figure 10.8 Work Breakdown Structure Using %EVG_WBS_CHART



The CPM procedure can be used as follows to generate a schedule based on the precedence and hierarchical constraints (see Chapter 4 for more details).

```
proc cpm data=softin
         out=software
         interval=day
         date='01mar10'd;
   id description;
   project project / addwbs;
   activity activity;
   duration duration;
   successor succesr1 succesr2;
run;
```

The resulting schedule data set is manipulated to produce a data set containing the early and late schedules, shown in Figure 10.9. The WBS_CODE variable gives the project hierarchy. The E_START and E_FINISH variables represent the early schedule, and the L_START and L_FINISH variables give the late schedule.

```
proc sort data=software;
   by wbs_code;
run;
```

```
proc print data=software;
    title 'Planned Schedule';
    var activity wbs_code e_start e_finish l_start l_finish;
run;
```

Figure 10.9 Software Schedule SOFTWARE

Planned Schedule

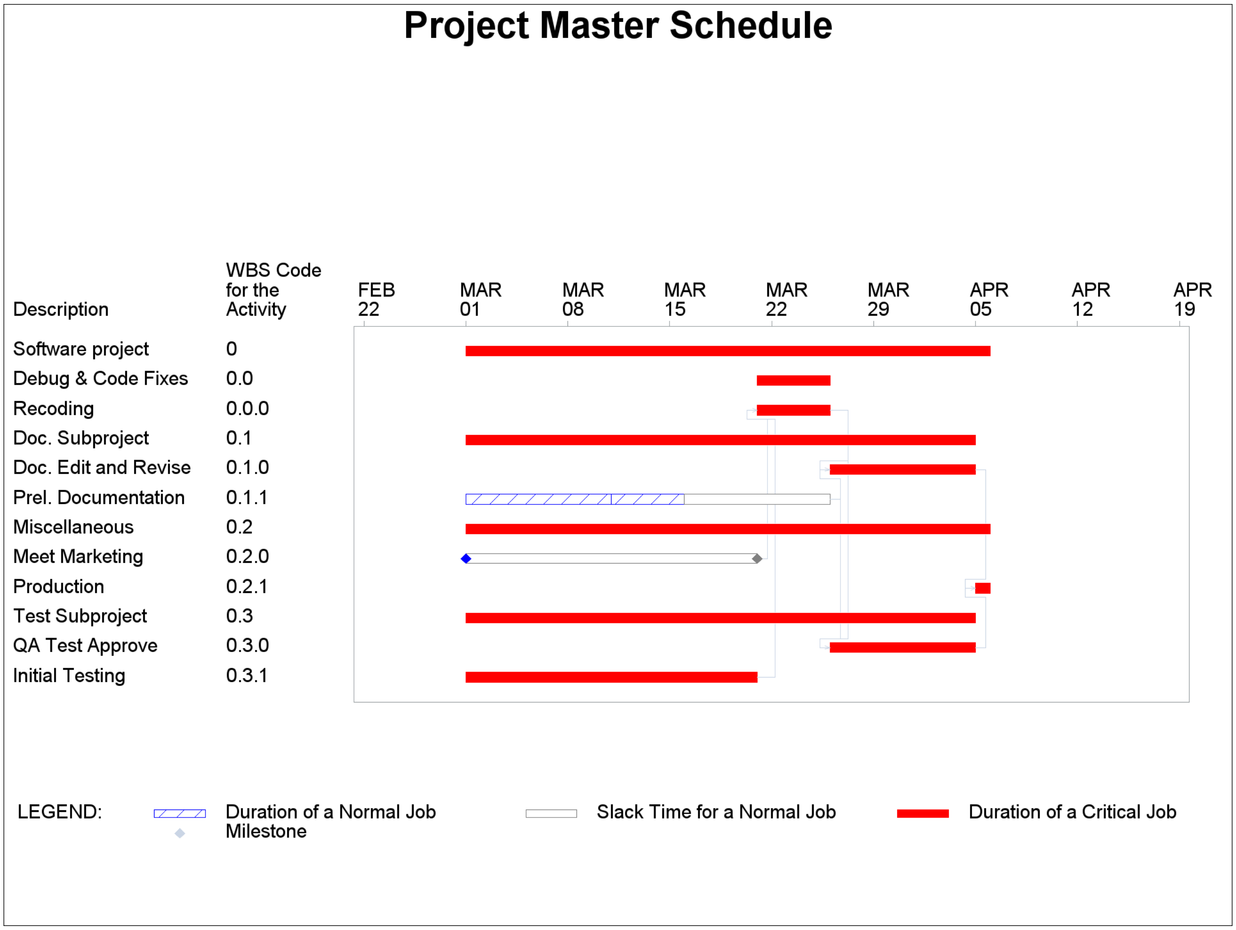
Obs	Activity	WBS_CODE	E_START	E_FINISH	L_START	L_FINISH
1	SWPROJ	0	01MAR10	05APR10	01MAR10	05APR10
2	DEBUG	0.0	21MAR10	25MAR10	21MAR10	25MAR10
3	RECODE	0.0.0	21MAR10	25MAR10	21MAR10	25MAR10
4	DOC	0.1	01MAR10	04APR10	11MAR10	04APR10
5	DOCEDREV	0.1.0	26MAR10	04APR10	26MAR10	04APR10
6	PRELDOC	0.1.1	01MAR10	15MAR10	11MAR10	25MAR10
7	MISC	0.2	01MAR10	05APR10	21MAR10	05APR10
8	MEETMKT	0.2.0	01MAR10	01MAR10	21MAR10	21MAR10
9	PROD	0.2.1	05APR10	05APR10	05APR10	05APR10
10	TEST	0.3	01MAR10	04APR10	01MAR10	04APR10
11	QATEST	0.3.0	26MAR10	04APR10	26MAR10	04APR10
12	TESTING	0.3.1	01MAR10	20MAR10	01MAR10	20MAR10

A Gantt chart of this schedule can be produced with the GANTT procedure using the following code. (See Chapter 8, for more details.)

```
goptions reset=symbol;
goptions reset=pattern;
pattern1 value=R1 color=B REPEAT=1;
pattern2 value=E color=GR REPEAT=1;
pattern3 value=S color=R REPEAT=1;
pattern4 value=X1 color=P REPEAT=1;
pattern5 value=E color=BR REPEAT=1;
pattern6 value=L1 color=GR REPEAT=1;
pattern7 value=R1 color=G REPEAT=1;
pattern8 value=X1 color=Y REPEAT=1;
pattern9 value=E color=B REPEAT=1;

title h=4pct 'Project Master Schedule';
title2 h=3pct;
proc gantt data=software;
   chart / mininterval=week pcompress
      height=1.3
      skip=2
      activity=activity
      nojobnum
      successor=(succesr1 succesr2)
      scale=10
      duration=duration;
      id description wbs_code;
run;
```

Figure 10.10 shows the output from this call to PROC GANTT.

Figure 10.10 Initial Schedule Using PROC GANTT



Now suppose the early schedule is designated as the baseline. For the sake of clarity, the E_START and E_FINISH variables have been renamed to START and FINISH, respectively. Also, the variables required for performing the earned value analysis are retained. The modified data set is shown in Figure 10.11.

```
data software;
   set software;
   keep project activity duration description
        e_start e_finish wbs_code succesr1 succesr2;
   rename e_start = Start
          e_finish = Finish;
   label wbs_code = 'WBS Code'
         e_start = 'Start'
         e_finish = 'Finish';
   if duration eq . then duration = proj_dur;
run;
```

Figure 10.11 Software Schedule SOFTWARE

Planned Schedule

Obs	Activity	WBS Code	Description	Duration	Start	Finish
1	SWPROJ	0	Software project	36	01MAR10	05APR10
2	DEBUG	0.0	Debug & Code Fixes	5	21MAR10	25MAR10
3	RECODE	0.0.0	Recoding	5	21MAR10	25MAR10
4	DOC	0.1	Doc. Subproject	35	01MAR10	04APR10
5	DOCEDREV	0.1.0	Doc. Edit and Revise	10	26MAR10	04APR10
6	PRELDOC	0.1.1	Prel. Documentation	15	01MAR10	15MAR10
7	MISC	0.2	Miscellaneous	36	01MAR10	05APR10
8	MEETMKT	0.2.0	Meet Marketing	0	01MAR10	01MAR10
9	PROD	0.2.1	Production	1	05APR10	05APR10
10	TEST	0.3	Test Subproject	35	01MAR10	04APR10
11	QATEST	0.3.0	QA Test Approve	10	26MAR10	04APR10
12	TESTING	0.3.1	Initial Testing	20	01MAR10	20MAR10

Assume also that each of the activities is budgeted to incur costs continuously at the rates shown in the RATES data set in Figure 10.12.

Figure 10.12 Software Budget RATES

Planned Rates

Obs	Activity	Rate
1	SWPROJ	5
2	RECODE	6
3	PRELDOC	4
4	DOCEDREV	4
5	MEETMKT	.
6	PROD	2
7	TEST	1
8	DOC	1
9	MISC	1
10	DEBUG	1
11	TESTING	3
12	QATEST	4

To determine the periodic budgeted costs, invoke the %EVA_PLANNED_VALUE macro as follows:

```
%eva_planned_value(
     plansched=software,       /* schedule data */
     activity=activity,
     start=start,
     finish=finish,
     duration=duration,
     budgetcost=rates,         /* cost data */
     rate=rate
);
```

The parameters used in this call to %EVA_PLANNED_VALUE are described in the following list. More details are given in the section "Syntax: Earned Value Management Macros" on page 809.

- PLANSCHED= identifies the data set that contains the planned schedule.
- ACTIVITY= specifies the activity variable in the PLANSCHED= and BUDGETCOST= data sets.
- START= specifies the start date or datetime variable in the PLANSCHED= data set.
- FINISH= specifies the finish date or datetime variable in the PLANSCHED= data set.
- DURATION= specifies the task duration variable in the PLANSCHED= data set.
- BUDGETCOST= identifies the data set containing the budgeted costs.
- RATE= specifies the cost rate variable in the BUDGETCOST= data set.

The output data set generated by this call to %EVA_PLANNED_VALUE is shown in Figure 10.13.

Figure 10.13 Periodic Planned Value Data Set Using %EVA_PLANNED_VALUE

Daily Planned Value

Obs	Date	PV Rate
1	01MAR10	15
2	02MAR10	15
3	03MAR10	15
4	04MAR10	15
5	05MAR10	15
6	06MAR10	15
7	07MAR10	15
8	08MAR10	15
9	09MAR10	15
10	10MAR10	15
11	11MAR10	15
12	12MAR10	15
13	13MAR10	15
14	14MAR10	15
15	15MAR10	15
16	16MAR10	11
17	17MAR10	11
18	18MAR10	11
19	19MAR10	11
20	20MAR10	11
21	21MAR10	15
22	22MAR10	15
23	23MAR10	15
24	24MAR10	15
25	25MAR10	15
26	26MAR10	16
27	27MAR10	16
28	28MAR10	16
29	29MAR10	16
30	30MAR10	16
31	31MAR10	16
32	01APR10	16
33	02APR10	16
34	03APR10	16
35	04APR10	16
36	05APR10	8

Note that the “PV Rate” column shows the daily Planned Value (or Budgeted Cost of Work Scheduled). This is the cost incurred each day according to the plan or budget.

In general, once a project is in progress it is subject to uncertainties, the impact of which can often only be approximated in the original plan. Such uncertainties can affect the project schedule, project cost, or both. From a schedule standpoint an activity may be delayed due to a multitude of factors—a required resource being unavailable, a worker becoming sick, a machine breaking down, adverse weather conditions, etc. From a cost standpoint a contractor may revise his/her original estimate, the cost of raw materials may increase due to external factors, a sick or disabled worker might necessitate the use of a more costly contractor to minimize schedule slippage, etc.

Figure 10.14 lists the status of completed activities, and those in progress, as of March 25, 2010.

Figure 10.14 Software Project Status SOFTACT

Actual Dates and Percentage Complete

Obs	Activity	Start	Finish	Percent
1	MEETMKT	01MAR10	01MAR10	100
2	PRELDOC	01MAR10	14MAR10	100
3	TESTING	01MAR10	.	80

Integrating this new information with the original schedule input data set gives the updated data set shown in Figure 10.15.

```
proc sql;
   create table softupd as
      select project, softin.activity, duration,
             start, finish, percent, succesr1,
             succesr2, description
         from softin left join softact
         on softin.activity = softact.activity
         order by 1, 2
   ;
quit;
```

Figure 10.15 Updated Software Schedule Input SOFTUPD

Schedule Input March 25

Obs	Project	Activity	Duration	Start	Finish	Percent	Succesr1	Succesr2
1		SWPROJ	.	.	.	.		
2	DEBUG	RECODE	5	.	.	.	DOCEDREV	QATEST
3	DOC	DOCEDREV	10	.	.	.	PROD	
4	DOC	PRELDOC	15	01MAR10	14MAR10	100	DOCEDREV	QATEST
5	MISC	MEETMKT	0	01MAR10	01MAR10	100	RECODE	
6	MISC	PROD	1	.	.	.		
7	SWPROJ	DEBUG	.	.	.	.		
8	SWPROJ	DOC	.	.	.	.		
9	SWPROJ	MISC	.	.	.	.		
10	SWPROJ	TEST	.	.	.	.		
11	TEST	QATEST	10	.	.	.	PROD	
12	TEST	TESTING	20	01MAR10	.	80	RECODE	

Using this updated data set, PROC CPM is then invoked to create a new schedule.

```
proc cpm data=softupd
         out=software25
         interval=day
         date='01mar10'd;
   id description percent;
   project project / addwbs;
   activity activity;
   duration duration;
   actual / as=start af=finish pctcomp=percent timenow='25MAR10'd;
   successor succesr1 succesr2;
run;

data software25;
   set software25;
   keep project activity e_start e_finish percent wbs_code duration;
   rename e_start = Start
          e_finish = Finish;
   label wbs_code = 'WBS Code'
         e_start = 'Start'
         e_finish = 'Finish';
run;
```

The resulting schedule is shown alongside the planned schedule in Figure 10.16.

Figure 10.16 Updated Software Schedule SOFTWARE25

Software Schedule as of March 25

Obs	Activity	WBS Code	Planned Start	Planned Finish	Start	Finish	Percent
1	SWPROJ	0	01MAR10	05APR10	01MAR10	15APR10	.
2	DEBUG	0.0	21MAR10	25MAR10	31MAR10	04APR10	.
3	RECODE	0.0.0	21MAR10	25MAR10	31MAR10	04APR10	.
4	DOC	0.1	01MAR10	04APR10	01MAR10	14APR10	.
5	DOCEDREV	0.1.0	26MAR10	04APR10	05APR10	14APR10	.
6	PRELDOC	0.1.1	01MAR10	15MAR10	01MAR10	14MAR10	100
7	MISC	0.2	01MAR10	05APR10	01MAR10	15APR10	.
8	MEETMKT	0.2.0	01MAR10	01MAR10	01MAR10	01MAR10	100
9	PROD	0.2.1	05APR10	05APR10	15APR10	15APR10	.
10	TEST	0.3	01MAR10	04APR10	01MAR10	14APR10	.
11	QATEST	0.3.0	26MAR10	04APR10	05APR10	14APR10	.
12	TESTING	0.3.1	01MAR10	20MAR10	01MAR10	30MAR10	80

Notice that the activity PRELDOC has completed one day ahead of schedule. Despite this encouraging bit of news, the project is showing signs of slipping. Only one other activity has completed as of the status date. The original plan called for three activities to have completed, with two additional activities completing at the end of the day. Instead, five activities are in progress. The finish date for the root parent activity, SWPROJ, shows the magnitude of the project slippage so far.

The updated cost rate information is given in Figure 10.17. Compared to the numbers listed in Figure 10.12 the rates for the PRELDOC and TESTING activities have increased, while the rate for the RECODE activity has decreased. Note that the original rates still apply for the activities that are not listed here.

Figure 10.17 Actual Software Costs RATES25

Actual Rates Day 25

Obs	Activity	Rate
1	PRELDOC	5
2	RECODE	5
3	TESTING	4

To compute the impact of schedule slippage and cost overruns, data sets SOFTWARE25 and RATES25 can be specified for the %EVA_EARNED_VALUE macro as shown in the following SAS code. It is assumed that the %EVA_PLANNED_VALUE macro has been run, in which case the planned duration and costs are implicitly carried over to the %EVA_EARNED_VALUE macro as shown in Figure 10.1. Also, as before, the E_START and E_FINISH variables have been renamed to START and FINISH, respectively.

```
%eva_earned_value(
    revisesched=software25, /* revised schedule */
    activity=activity,
    start=start,
    finish=finish,
    actualcost=rates25,     /* actual costs */
    rate=rate
);
```

The parameters used in this call to %EVA_EARNED_VALUE are described briefly in the following list. More details are given in the section "Syntax: Earned Value Management Macros" on page 809.

- REVISESCHED= identifies the data set that contains the revised schedule.
- ACTIVITY= specifies the activity variable in the REVISESCHED= and ACTUALCOST= data sets.
- START= specifies the start date or datetime variable in the REVISESCHED= data set.
- FINISH= specifies the finish date or datetime variable in the REVISESCHED= data set.
- ACTUALCOST= identifies the data set containing the revised cost. Activities not included are assigned the budgeted cost.
- RATE= specifies the cost rate variable in the ACTUALCOST= data set.

The variable names specified for the ACTIVITY=, START=, FINISH=, and RATE= parameters must be the same, respectively, as those previously specified for the %EVA_PLANNED_VALUE macro. This is to enable the later use of the %EVA_TASK_METRICS and %EVG_GANTT_CHART macros. The output generated by this call to %EVA_EARNED_VALUE is shown in Figure 10.18.

Figure 10.18 Periodic Earned Value Data Set Using %EVA_EARNED_VALUE

Daily Earned Value and Revised Cost

Obs	Date	EV Rate	AC Rate
1	01MAR10	12.5369	17
2	02MAR10	12.5369	17
3	03MAR10	12.5369	17
4	04MAR10	12.5369	17
5	05MAR10	12.5369	17
6	06MAR10	12.5369	17
7	07MAR10	12.5369	17
8	08MAR10	12.5369	17
9	09MAR10	12.5369	17
10	10MAR10	12.5369	17
11	11MAR10	12.5369	17
12	12MAR10	12.5369	17
13	13MAR10	12.5369	17
14	14MAR10	12.5369	17
15	15MAR10	8.2512	12
16	16MAR10	8.2512	12
17	17MAR10	8.2512	12
18	18MAR10	8.2512	12
19	19MAR10	8.2512	12
20	20MAR10	8.2512	12
21	21MAR10	8.2512	12
22	22MAR10	8.2512	12
23	23MAR10	8.2512	12
24	24MAR10	8.2512	12
25	25MAR10	8.2512	12
26	26MAR10	8.2512	12
27	27MAR10	8.2512	12
28	28MAR10	8.2512	12
29	29MAR10	8.2512	12
30	30MAR10	8.2512	12
31	31MAR10	13.2512	14
32	01APR10	13.2512	14
33	02APR10	13.2512	14
34	03APR10	13.2512	14
35	04APR10	13.2512	14
36	05APR10	14.2512	16
37	06APR10	14.2512	16
38	07APR10	14.2512	16
39	08APR10	14.2512	16
40	09APR10	14.2512	16
41	10APR10	14.2512	16
42	11APR10	14.2512	16
43	12APR10	14.2512	16
44	13APR10	14.2512	16
45	14APR10	14.2512	16
46	15APR10	6.6957	8

The “EV Rate” column shows the daily Earned Value (or Budgeted Cost of Work Performed). This is the budgeted cost for the work that was actually accomplished each day. Up to the status date (TIMENOW) of March 25, 2010, the “AC Rate” column depicts the daily Actual Cost (or Actual Cost of Work Performed). After this date, this column represents estimated costs. For this estimate, it is assumed that activities that are in progress at the status date continue at the same cost rate, rather than reverting to the budgeted cost rate. This assumption ultimately yields the revised Estimate At Completion (EAC_{rev}). Figure 10.19 shows the daily Planned Value, Earned Value, and Actual Cost together. The disparity in the number of observations (47 here versus the former 37 for the %EVA_PLANNED_VALUE macro) reflects a schedule slippage of 10 days.

Figure 10.19 Periodic Planned Value, Earned Value, and Actual Cost

Daily Planned Value, Earned Value, and Revised Cost

Obs	DATE	PV Rate	EV Rate	AC Rate
1	01MAR10	15	12.5369	17
2	02MAR10	15	12.5369	17
3	03MAR10	15	12.5369	17
4	04MAR10	15	12.5369	17
5	05MAR10	15	12.5369	17
6	06MAR10	15	12.5369	17
7	07MAR10	15	12.5369	17
8	08MAR10	15	12.5369	17
9	09MAR10	15	12.5369	17
10	10MAR10	15	12.5369	17
11	11MAR10	15	12.5369	17
12	12MAR10	15	12.5369	17
13	13MAR10	15	12.5369	17
14	14MAR10	15	12.5369	17
15	15MAR10	15	8.2512	12
16	16MAR10	11	8.2512	12
17	17MAR10	11	8.2512	12
18	18MAR10	11	8.2512	12
19	19MAR10	11	8.2512	12
20	20MAR10	11	8.2512	12
21	21MAR10	15	8.2512	12
22	22MAR10	15	8.2512	12
23	23MAR10	15	8.2512	12
24	24MAR10	15	8.2512	12
25	25MAR10	15	8.2512	12
26	26MAR10	16	8.2512	12
27	27MAR10	16	8.2512	12
28	28MAR10	16	8.2512	12
29	29MAR10	16	8.2512	12
30	30MAR10	16	8.2512	12
31	31MAR10	16	13.2512	14
32	01APR10	16	13.2512	14
33	02APR10	16	13.2512	14
34	03APR10	16	13.2512	14
35	04APR10	16	13.2512	14
36	05APR10	8	14.2512	16
37	06APR10	.	14.2512	16
38	07APR10	.	14.2512	16
39	08APR10	.	14.2512	16
40	09APR10	.	14.2512	16
41	10APR10	.	14.2512	16
42	11APR10	.	14.2512	16
43	12APR10	.	14.2512	16
44	13APR10	.	14.2512	16
45	14APR10	.	14.2512	16
46	15APR10	.	6.6957	8

Results

The earned value metrics can now be computed using the following code:

```
%eva_metrics(timenow='25MAR10'd);
```

It is assumed that the %EVA_PLANNED_VALUE and %EVA_EARNED_VALUE macros have been run and that the default output data sets, PV and EV, were used. This enables the %EVA_METRICS macro to implicitly use those data sets as input (see Figure 10.1). If the default data set names were not used, you must specify the correct names using the PV= and EV= options.

The TIMENOW= parameter specifies the date or datetime to which the updated schedule and rates apply.

Figure 10.20 shows the output from the %EVA_METRICS macro.

Figure 10.20 Earned Value Summary Metrics Using %EVA_METRICS

Earned Value Analysis
as of March 25, 2010

Metric	Value
Percent Complete	50.91
PV (Planned Value)	355.00
EV (Earned Value)	266.28
AC (Actual Cost)	370.00
CV (Cost Variance)	-103.72
CV%	-38.95
SV (Schedule Variance)	-88.72
SV%	-24.99
CPI (Cost Performance Index)	0.72
SPI (Schedule Performance Index)	0.75
BAC (Budget At Completion)	523.00
EAC (Revised Estimate At Completion)	668.00
EAC (Overrun to Date)	626.72
EAC (Cumulative CPI)	726.72
EAC (Cumulative CPI X SPI)	845.57
ETC (Estimate To Complete)*	356.72
VAC (Variance At Completion)*	-203.72
VAC%*	-38.95
TCPI (BAC) (To-Complete Performance Index)	1.68
TCPI (EAC) (To-Complete Performance Index)*	0.72

*** The CPI form of the EAC is used.**

The metrics in Figure 10.20 reflect the condition of the project at the status date (TIMENOW). "Percent Complete" is the percentage of the planned work that has been accomplished. Values for this entry range from 0 to 100. PV is the Planned Value, or Budgeted Cost of Work Scheduled (BCWS). This is the work that was projected to have been completed. EV is the Earned Value, or Budgeted Cost of Work Performed (BCWP). This measure represents the value of work completed so far, in terms of budgeted cost. AC is the Actual Cost of Work Performed (ACWP), and represents the actual cost of the work so far.

At first glance, one might compare the Planned Value of $355 to the Actual Cost of $370, and conclude that costs are nearly on track. However, the Earned Value figure of $266.28 puts these numbers in the proper perspective. This project is careening out of control with respect to schedule and costs.

CV and SV are the Cost and Schedule Variance, respectively. CV is the earned value less the amount spent. SV is the earned value less the planned value. CPI and SPI are the Cost and Schedule Performance indices, respectively, and are analogous to the CV and SV. The CPI and SPI are expressed as ratios, rather than differences. The value of 0.72 for CPI indicates a cost overrun situation in which the project has earned 72 cents for every dollar spent. The value of 0.75 for SPI foretells a late project, as only 75% of the planned work has been accomplished on the status date. BAC is the Budget at Completion and is the total budgeted cost of the project. EAC is the Estimate at Completion, and represents the projected total cost, given the current state of the project. For the first of the EAC metrics, EAC_{rev}, it is assumed that future work is performed at the revised rates. EAC_{rev} is computed by accumulating the "AC Rate" data found in the output data set from the %EVA_EARNED_VALUE macro. For EAC_{OTD}, the overrun to date estimate, it is assumed that future work will be done at the budgeted rate, which is typically not very realistic. The next two estimates take into account the performance of the project so far. EAC_{CPI}, or the cumulative CPI EAC, is widely regarded as one of the more accurate and reliable estimates, and is frequently used to provide a quick forecast of the project costs. By default, the EAC_{CPI} is used to compute the Variance at Completion (VAC) and one of the To-Complete Performance Index (TCPI) formulations, to follow. The cumulative CPI-times-SPI EAC, denoted $\text{EAC}_{\text{CPI}\times\text{SPI}}$, is often used to provide a high-end estimate of the project costs. ETC is the Estimate to Complete, or a projection of remaining costs. The VAC represents the financial impact of the expenditures to date on the original budget.

The TCPI is the cost performance factor that must be achieved in order to complete the remaining work using the available funds. The available funds can be defined to be either the remaining budget $(\text{BAC} - \text{AC})$ or remaining costs $(\text{EAC} - \text{AC})$. For example, the TCPI_{BAC} of 1.68 indicates that the project needs to earn 1.68 dollars for each dollar spent to stay within the budget. The TCPI_{EAC} is a far lower value (0.72) because the revised costs are much larger than the budgeted costs for the remaining work. See Table 10.11, Table 10.12, and Table 10.13 for formula expressions of the preceding metrics.

Figure 10.21 gives a listing of the earned value summary statistics data set produced by the %EVA_METRICS macro. This data set is used to generate the %EVA_METRICS report as well as some of the charts to be described in later sections.

Figure 10.21 Summary Earned Value Data Set Using %EVA_METRICS

Earned Value Summary Statistics

Obs	Name	Label	March 25, 2010
1	_pctcomp_	Percent Complete	50.914
2	_pv_	PV (Planned Value)	355.000
3	_ev_	EV (Earned Value)	266.280
4	_ac_	AC (Actual Cost)	370.000
5	_cv_	CV (Cost Variance)	-103.720
6	_cvp_	CV%	-38.951
7	_sv_	SV (Schedule Variance)	-88.720
8	_svp_	SV%	-24.991
9	_cpi_	CPI (Cost Performance Index)	0.720
10	_spi_	SPI (Schedule Performance Index)	0.750
11	_bac_	BAC (Budget At Completion)	523.000
12	_eacrev_	EAC (Revised Estimate At Completion)	668.000
13	_eacotd_	EAC (Overrun to Date)	626.720
14	_eaccpi_	EAC (Cumulative CPI)	726.716
15	_eaccpispi_	EAC (Cumulative CPI X SPI)	845.567
16	_etc_	ETC (Estimate To Complete)*	356.716
17	_vac_	VAC (Variance At Completion)*	-203.716
18	_vacp_	VAC%*	-38.951
19	_tcpi_	TCPI (BAC) (To-Complete Performance Index)	1.678
20	_etcpi_	TCPI (EAC) (To-Complete Performance Index)*	0.720

The %EVA_METRICS macro also produces the cumulative periodic earned value data set shown in Figure 10.22. This data set contains the cumulative planned value, earned value, and actual and revised costs, together with the associated variances and performance indices.

Figure 10.22 Cumulative Earned Value Data Set Using %EVA_METRICS

Earned Value Cumulative Periodic Metrics
Values, Costs, Variances and Indices

Obs	Date	PV	EV	AC	Revised Cost	CV	SV	CPI	SPI
1	01MAR10	15	12.537	17	17	-4.463	-2.4631	0.73747	0.83579
2	02MAR10	30	25.074	34	34	-8.926	-4.9262	0.73747	0.83579
3	03MAR10	45	37.611	51	51	-13.389	-7.3892	0.73747	0.83579
4	04MAR10	60	50.148	68	68	-17.852	-9.8523	0.73747	0.83579
5	05MAR10	75	62.685	85	85	-22.315	-12.3154	0.73747	0.83579
6	06MAR10	90	75.222	102	102	-26.778	-14.7785	0.73747	0.83579
7	07MAR10	105	87.758	119	119	-31.242	-17.2415	0.73747	0.83579
8	08MAR10	120	100.295	136	136	-35.705	-19.7046	0.73747	0.83579
9	09MAR10	135	112.832	153	153	-40.168	-22.1677	0.73747	0.83579
10	10MAR10	150	125.369	170	170	-44.631	-24.6308	0.73747	0.83579
11	11MAR10	165	137.906	187	187	-49.094	-27.0939	0.73747	0.83579
12	12MAR10	180	150.443	204	204	-53.557	-29.5569	0.73747	0.83579
13	13MAR10	195	162.980	221	221	-58.020	-32.0200	0.73747	0.83579
14	14MAR10	210	175.517	238	238	-62.483	-34.4831	0.73747	0.83579
15	15MAR10	225	183.768	250	250	-66.232	-41.2319	0.73507	0.81675
16	16MAR10	236	192.019	262	262	-69.981	-43.9807	0.73290	0.81364
17	17MAR10	247	200.271	274	274	-73.729	-46.7295	0.73091	0.81081
18	18MAR10	258	208.522	286	286	-77.478	-49.4783	0.72910	0.80822
19	19MAR10	269	216.773	298	298	-81.227	-52.2271	0.72743	0.80585
20	20MAR10	280	225.024	310	310	-84.976	-54.9758	0.72588	0.80366
21	21MAR10	295	233.275	322	322	-88.725	-61.7246	0.72446	0.79076
22	22MAR10	310	241.527	334	334	-92.473	-68.4734	0.72313	0.77912
23	23MAR10	325	249.778	346	346	-96.222	-75.2222	0.72190	0.76855
24	24MAR10	340	258.029	358	358	-99.971	-81.9710	0.72075	0.75891
25	25MAR10	355	266.280	370	370	-103.720	-88.7198	0.71968	0.75009
26	26MAR10	371	.	.	382	.	.	.	.
27	27MAR10	387	.	.	394	.	.	.	.
28	28MAR10	403	.	.	406	.	.	.	.
29	29MAR10	419	.	.	418	.	.	.	.
30	30MAR10	435	.	.	430	.	.	.	.
31	31MAR10	451	.	.	444	.	.	.	.
32	01APR10	467	.	.	458	.	.	.	.
33	02APR10	483	.	.	472	.	.	.	.
34	03APR10	499	.	.	486	.	.	.	.
35	04APR10	515	.	.	500	.	.	.	.
36	05APR10	523	.	.	516	.	.	.	.
37	06APR10	.	.	.	532	.	.	.	.
38	07APR10	.	.	.	548	.	.	.	.
39	08APR10	.	.	.	564	.	.	.	.
40	09APR10	.	.	.	580	.	.	.	.
41	10APR10	.	.	.	596	.	.	.	.
42	11APR10	.	.	.	612	.	.	.	.
43	12APR10	.	.	.	628	.	.	.	.
44	13APR10	.	.	.	644	.	.	.	.

Figure 10.22 *continued*

Earned Value Cumulative Periodic Metrics
Values, Costs, Variances and Indices

Obs	Date	PV	EV	AC	Revised Cost	CV	SV	CPI	SPI
45	14APR10	.	.	.	660	.	.	.	.
46	15APR10	.	.	.	668	.	.	.	.

The cost and schedule variance by activity can now be produced using the following call to %EVA_TASK_METRICS.

```
%eva_task_metrics(
    plansched=software,
    revisesched=software25,
    activity=activity,
    start=start,
    finish=finish,
    budgetcost=rates,
    actualcost=rates25,
    rate=rate,
    aggregate=Y,
    timenow='25MAR10'd
);
```

The parameters used in this call to %EVA_TASK_METRICS are described briefly in the following list. More details are given in the section "Syntax: Earned Value Management Macros" on page 809.

- PLANSCHED= identifies the data set that contains the planned schedule.
- REVISESCHED= identifies the data set that contains the updated schedule.
- ACTIVITY= specifies the activity variable in the PLANSCHED= and REVISESCHED= data sets.
- START= specifies the start date or datetime variable in the PLANSCHED= and REVISESCHED= data sets.
- FINISH= specifies the finish date or datetime variable in the PLANSCHED= and REVISESCHED= data sets.
- BUDGETCOST= identifies the data set containing the budgeted cost.
- ACTUALCOST= identifies the data set containing the revised cost.
- RATE= specifies the cost rate variable in the BUDGETCOST= and ACTUALCOST= data sets.
- AGGREGATE= indicates whether or not to roll up values along the project hierarchy; a value of Y indicates that aggregation is to be performed.
- TIMENOW= specifies the date or datetime of the updated schedule and costs.

Note that the variable names specified for the ACTIVITY=, START=, FINISH=, and RATE= parameters must be the same, respectively, as those previously specified for the %EVA_PLANNED_VALUE and %EVA_EARNED_VALUE macros.

The output produced by this invocation of the %EVA_TASK_METRICS macro is shown in Figure 10.23.

Figure 10.23 Cost and Schedule Variance by Activity Using %EVA_TASK_METRICS

Earned Value Analysis by Activity
as of March 25, 2010

Obs	Activity	WBS Code	PV	EV	AC	CV	CV%	SV	SV%	CPI	SPI
1	SWPROJ	0	355.00	266.28	370.00	-103.72	-38.95	-88.72	-24.99	0.72	0.75
2	DEBUG	0.0	35.00	0.00	0.00	0.00	0.00	-35.00	-100.00	.	0.00
3	RECODE	0.0.0	30.00	0.00	0.00	0.00	0.00	-30.00	-100.00	.	0.00
4	DOC	0.1	85.00	79.44	95.00	-15.56	-19.58	-5.56	-6.54	0.84	0.93
5	DOCEDREV	0.1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.	.
6	PRELDOC	0.1.1	60.00	60.00	70.00	-10.00	-16.67	0.00	0.00	0.86	1.00
7	MISC	0.2	25.00	19.57	25.00	-5.43	-27.78	-5.43	-21.74	0.78	0.78
8	MEETMKT	0.2.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.	.
9	PROD	0.2.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.	.
10	TEST	0.3	85.00	69.44	125.00	-55.56	-80.00	-15.56	-18.30	0.56	0.82
11	QATEST	0.3.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.	.
12	TESTING	0.3.1	60.00	50.00	100.00	-50.00	-100.00	-10.00	-16.67	0.50	0.83

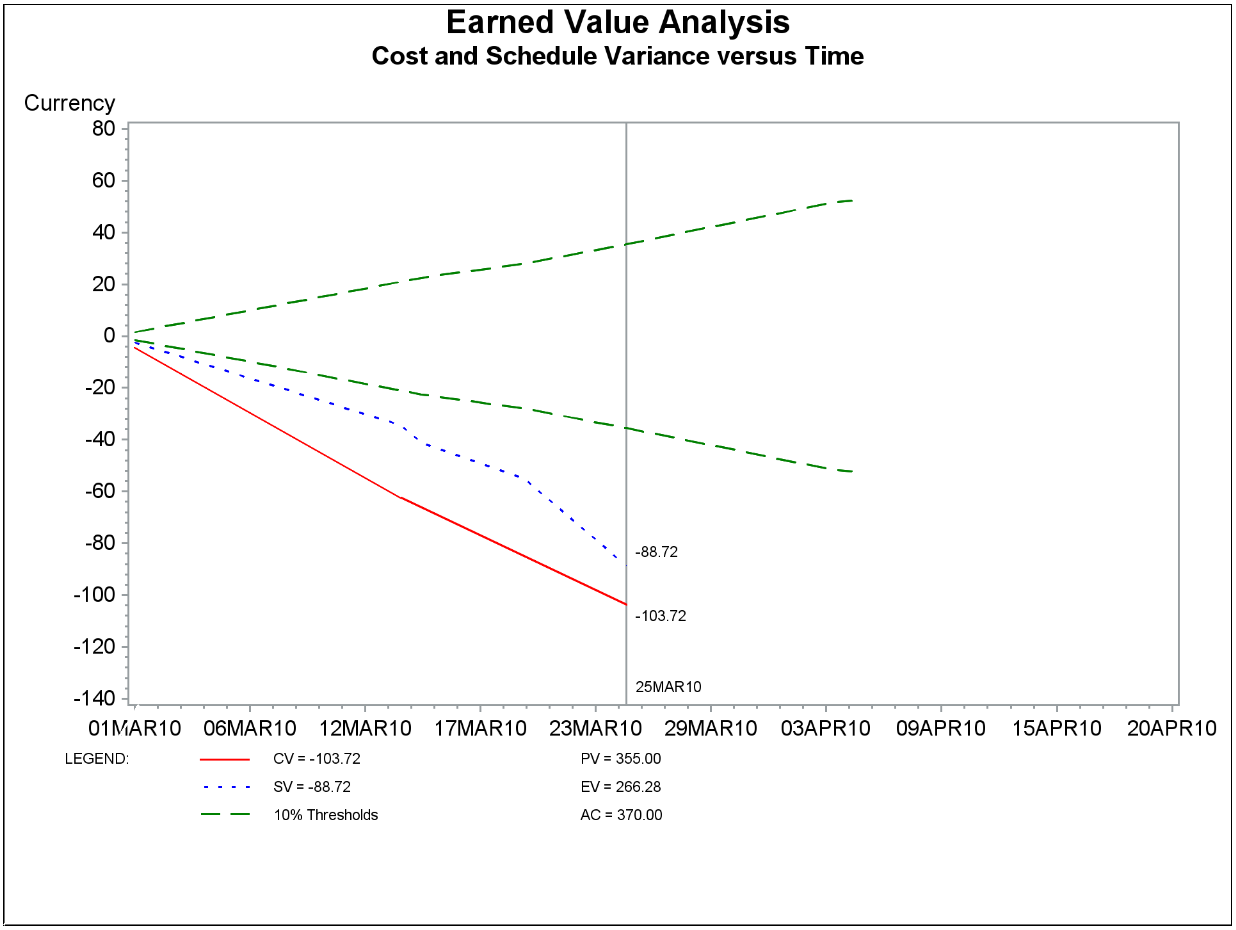
Note that by specifying Y for the AGGREGATE= parameter, the values have been rolled up along the project hierarchy; otherwise, the values would represent the corresponding activity only.

Charts

In Figure 10.24, cost and schedule variance are plotted against time using a default invocation of the %EVG_VARIANCE_PLOT macro. Notice that the variances lie outside the plus or minus 10% Planned Value thresholds, which is a cause for concern.

```
%evg_variance_plot;
```

Figure 10.24 CV and SV versus Time Using %EVG_VARIANCE_PLOT



Although not apparent from the preceding chart, as any project nears completion its schedule variance tends toward zero. This is because the earned value eventually converges to the planned value.

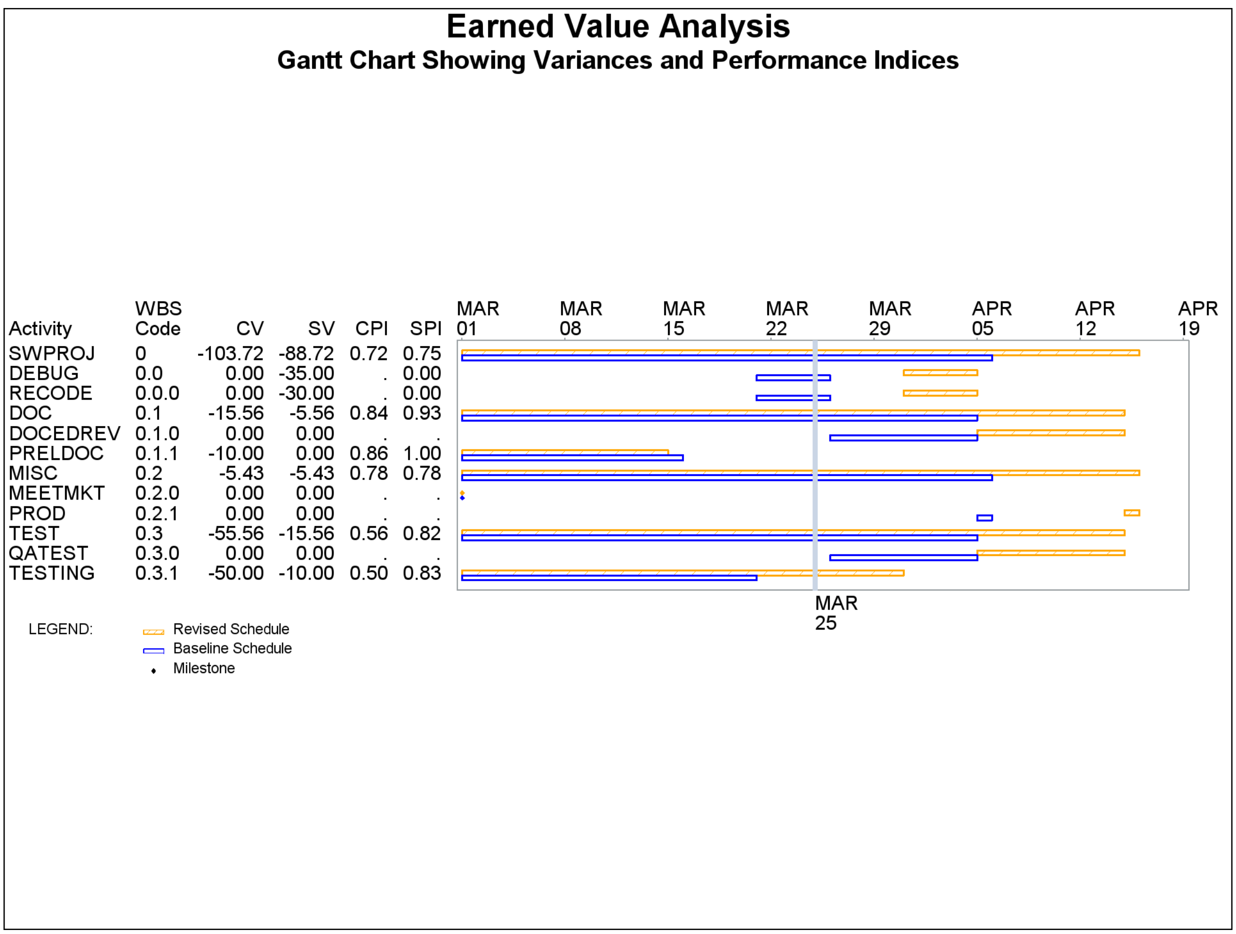
Note that the %EVA_METRICS macro must have been called prior to this invocation in order to generate the METRICS= data set.

The cost and schedule variance can also be displayed by activity on a Gantt chart with the %EVG_GANTT_CHART macro, as follows.

```
%evg_gantt_chart(
          plansched=software,
          revisesched=software25,
          duration=duration,
          start=start,
          finish=finish,
          activity=activity,
          timenow='25MAR10'd,
          id=wbs cv sv cpi spi,
          height=3,
          scale=20
);
```

Figure 10.25 depicts cost and schedule variance, in addition to the cost and schedule performance indices, with a Gantt-style schedule. Note that two bars are used to represent each activity. The top bar is the revised schedule and the bottom bar is the baseline schedule.

Figure 10.25 CV and SV by Task Using %EVG_GANTT_CHART



The parameters used in this call to %EVG_GANTT_CHART are described briefly in the following list. More details are given in the section "Syntax: Earned Value Management Macros" on page 809.

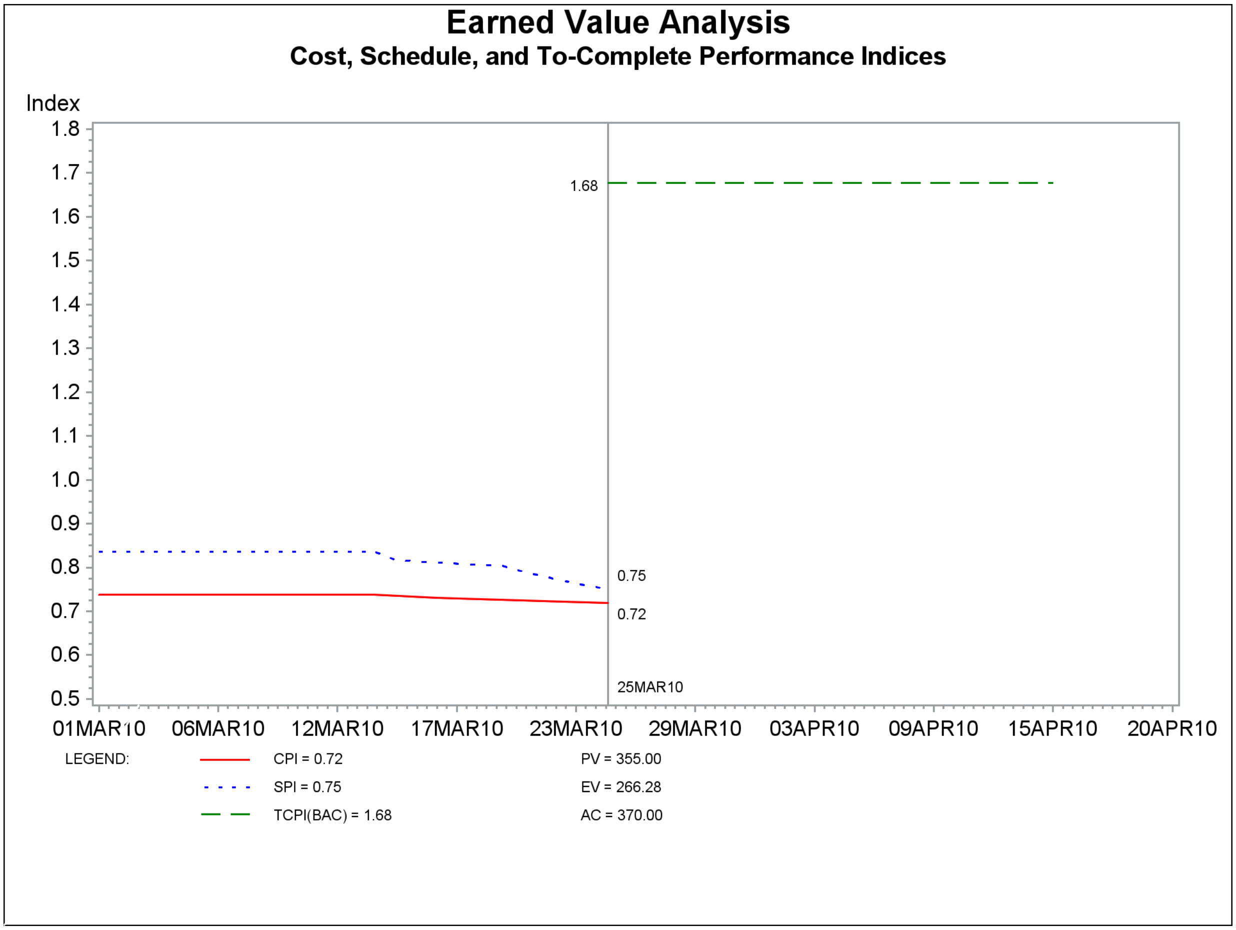
- PLANSCHED= identifies the data set containing the planned schedule.
- REVISESCHED= identifies the data set that contains the updated schedule.
- ACTIVITY= specifies the activity variable in the PLANSCHED= and REVISESCHED= data sets.
- START= specifies the start date or datetime variable in the PLANSCHED= and REVISESCHED= data sets.
- FINISH= specifies the finish date or datetime variable in the PLANSCHED= and REVISESCHED= data sets.
- DURATION= specifies a duration variable in the PLANSCHED= or REVISESCHED= data set. If the variable is present in both data sets, the REVISESCHED= data set is chosen as the source for the variable values. This variable is only used for differentiating milestones from single day tasks.
- TIMENOW= specifies the date or datetime of the updated schedule and costs.
- ID= specifies the variable(s) to be included in the Gantt chart.
- HEIGHT= specifies a height factor for all text in PROC GANTT. (See the HEIGHT= option for the CHART statement of PROC GANTT (Chapter 8) for more details.)
- SCALE= specifies the relative size adjustment of the data columns and chart. (See the SCALE= option for the CHART statement of PROC GANTT (Chapter 8) for more details.)

This macro relies on the output data set (named TASKMETS by default) that is generated by %EVA_TASK_METRICS. The TASKMETRICS= parameter is used by both %EVA_TASK_METRICS and %EVG_GANTT_CHART to specify a different data set name. More details are given in the section "Syntax: Earned Value Management Macros" on page 809.

The %EVG_INDEX_PLOT macro is used to generate a line plot of the cost and schedule performance indices, along with the to-complete performance index. The output generated by a default invocation is shown in Figure 10.26. Recall that CPI is the ratio of earned value to actual cost, and SPI is the ratio of earned value to planned value. Also, $\mathrm{TCPI}_{\mathrm{BAC}}$ is the ratio of work remaining to the remaining budget.

```
%evg_index_plot;
```

Figure 10.26 CPI, SPI, and TCPI Using %EVG_INDEX_PLOT



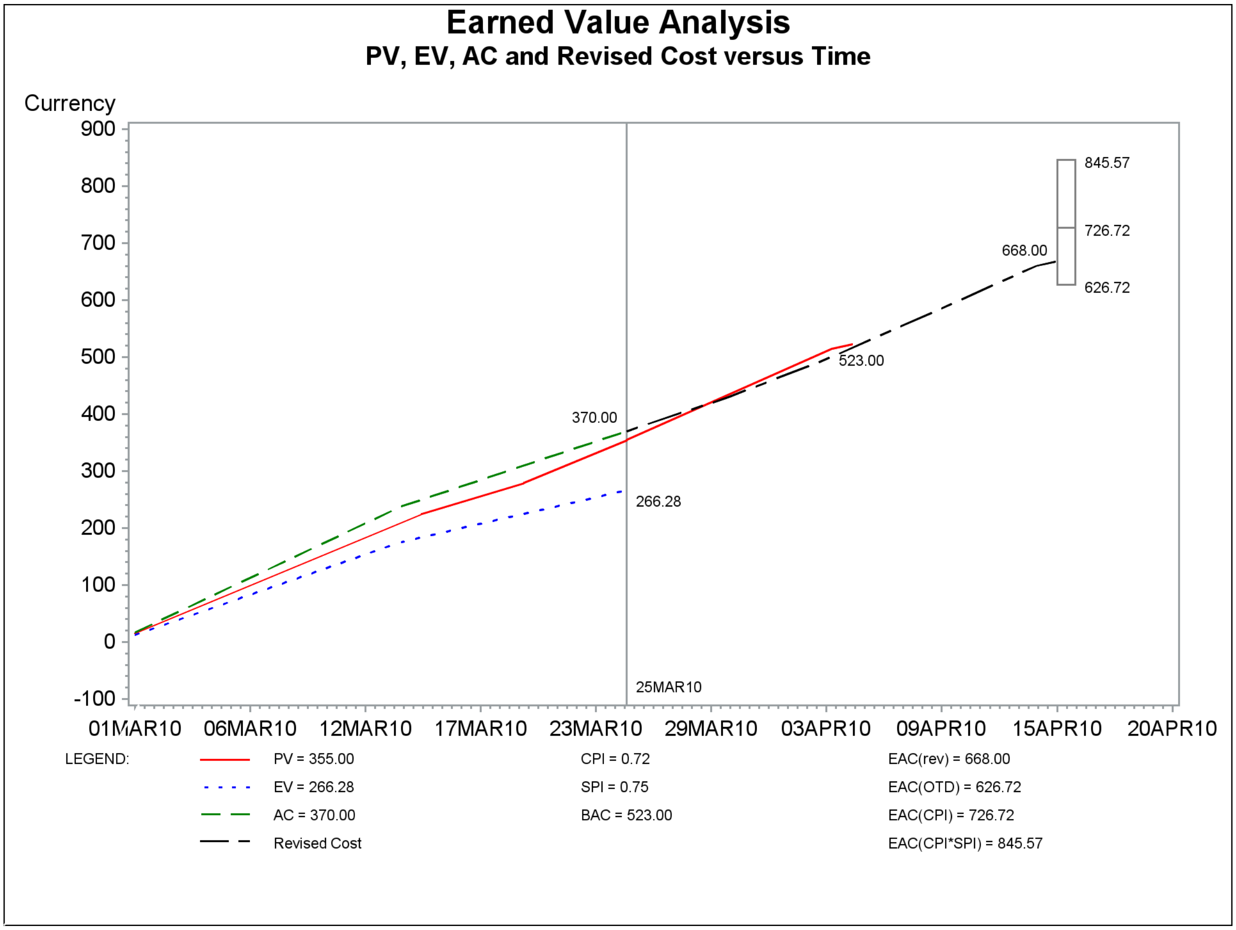
The CPI and SPI plots show a poor cost and schedule performance. Therefore the $\mathrm{TCPI}_{\mathrm{BAC}}$, an indicator of the performance required to get back on track, is relatively high.

Note that the %EVA_METRICS macro must have been called prior to this invocation in order to generate the METRICS= and SUMMARY= data sets.

The %EVG_COST_PLOT macro is used to plot the Budgeted Cost of Work Scheduled (or Planned Value), Budgeted Cost of Work Performed (or Earned Value), Actual Cost of Work Performed (or simply Actual Cost), and revised Estimate at Completion ($\mathrm{EAC}_{\mathrm{rev}}$) against time. The output from this macro is shown in Figure 10.27.

```
%evg_cost_plot;
```

Figure 10.27 PV, EV, AC, and $\mathrm{EAC}_{\mathrm{rev}}$ versus Time Using %EVG_COST_PLOT



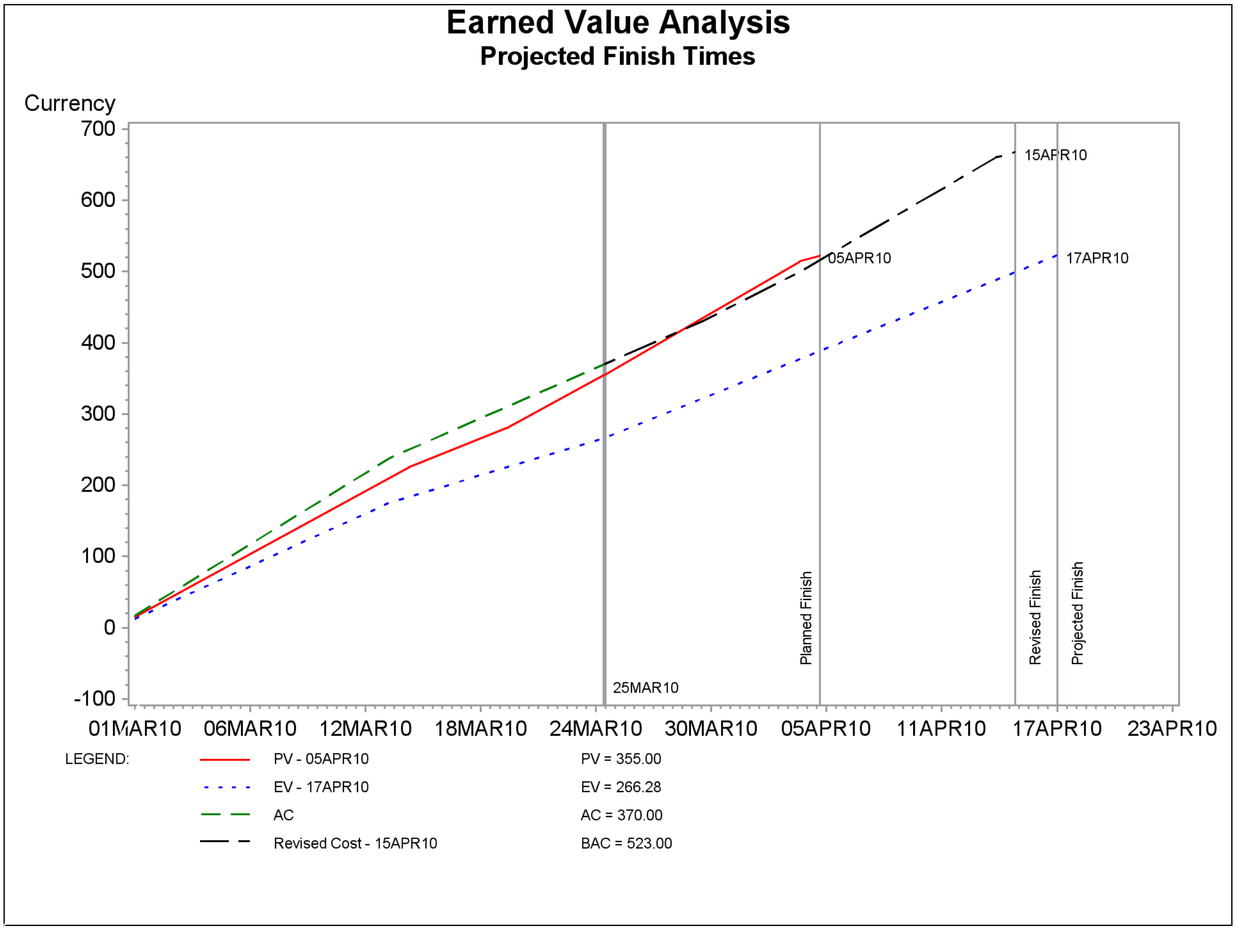
The $\mathrm{EAC}_{\mathrm{rev}}$ value of $668 depicted in Figure 10.27 is the sum of the actual costs so far and the revised cost of the future work. In other words, it is the sum of the "AC Rate" column in the output data set of the %EVA_EARNED_VALUE macro. With the optimistic $\mathrm{EAC}_{\mathrm{OTD}}$ value of $626.72, it is assumed that the remaining work will be completed according to the original budget. A more realistic value of $726.72 is shown for the $\mathrm{EAC}_{\mathrm{CPI}}$. Finally, the "upper bound" estimate of $845.57 is given by the $\mathrm{EAC}_{\mathrm{CPI}\times\mathrm{SPI}}$.

Note that the %EVA_METRICS macro must have been called prior to this invocation in order to generate the METRICS= and SUMMARY= data sets.

The %EVG_SCHEDULE_PLOT macro can be used to show planned, revised, and projected completion dates as pictured in Figure 10.28.

```
%evg_schedule_plot;
```

Figure 10.28 Projected Completion Dates Using %EVG_SCHEDULE_PLOT



In Figure 10.28, the current date (25MAR10) and planned end date (05APR10) are marked with vertical lines, as is the completion date according to the revised schedule (15APR10). The rightmost vertical line marks the projected finish time based upon the rate at which earned value has accumulated so far. This slope is used to extend the current earned value to the Budget at Completion (BAC) value of $523, providing a prediction of the completion date.

Note that the %EVA_METRICS macro must have been called prior to this invocation in order to generate the METRICS= data set.

Using the software schedule data set as input, the %EVG_WBS_CHART macro can be utilized to produce the Work Breakdown Structure seen in Figure 10.8.

Syntax: Earned Value Management Macros

Analysis

The following macros are available for earned value analysis.

%EVA_PLANNED_VALUE *(parameters)* **;**
%EVA_EARNED_VALUE *(parameters)* **;**
%EVA_METRICS *< (parameters) >* **;**
%EVA_TASK_METRICS *(parameters)* **;**

This section provides syntactical information for each of these macros. The macros are described in further detail in the section "Analysis" on page 827.

The role of each macro is illustrated in Figure 10.29. Each arrow entering a macro is associated with a *required* parameter specifying an input SAS data set. Each arrow leaving a macro is associated with an *optional* parameter specifying an output SAS data set. If an optional parameter does not receive an argument, then a default name is given to the output data set.

%EVA_PLANNED_VALUE

This macro is used to produce the periodic planned value.

%EVA_PLANNED_VALUE *(parameters)* **;**

Required Parameters

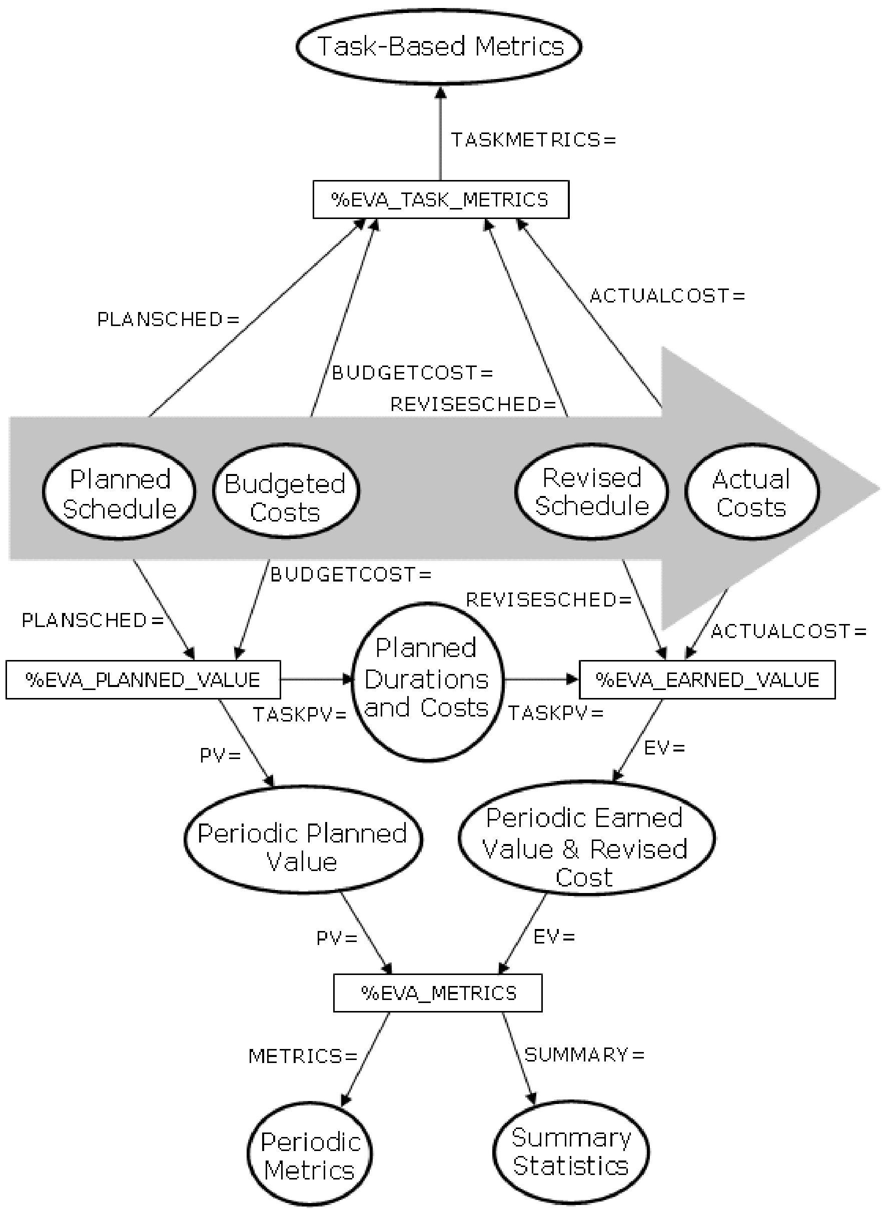
ACTIVITY=*variable*
specifies the activity variable in the PLANSCHED= and BUDGETCOST= data sets.

BUDGETCOST=*SAS-data-set*
PLANCOST=*SAS-data-set*
identifies the input data set containing the budgeted cost rates for each activity.

PLANSCHED=*SAS-data-set*
identifies the input data set containing the planned schedule.

START=*variable*
specifies the start date or datetime variable in the PLANSCHED= data set.

Figure 10.29 Analysis Macro Data Flow



Optional Parameters

CALENDAR=*SAS-data-set*

identifies the calendar input data set. (See the CALENDAR= option in Chapter 4 for more details.)

COST=*variable*

specifies the cost variable in the BUDGETCOST= data set. This is the total of the start and finish costs associated with the given task. At least COST= or RATE= must be specified.

DURATION=*variable*

specifies the task duration variable in the PLANSCHED= data set. At least DURATION= or FINISH= must be specified. If both parameters are specified, the FINISH= variable value overrides a nonzero DURATION= variable value; i.e., duration is only used to indicate a milestone task.

FINISH=*variable*

specifies the finish date or datetime variable in the PLANSCHED= data set. At least FINISH= or DURATION= must be specified. If both parameters are specified, the FINISH= variable value overrides a nonzero DURATION= variable value; i.e., duration is only used to indicate a milestone task.

HOLIDATA=*SAS-data-set*

identifies the input data set that contains the holiday schedule. (See the HOLIDATA= option in Chapter 4 for more details.)

HOLIDAY=*statement*

specifies the HOLIDAY statement to use for PROC CPM. For example:

```
%eva_planned_value(\ldots, holiday=holiday start / holifin=finish, ...);
```

(See the HOLIDAY statement in Chapter 4 for more details.)

INTERVAL=*interval*

specifies the units to use for the DURATION= variable values. (See the INTERVAL= option in Chapter 4 for more details.)

MAXOBS=*max*

sets the limit on the number of allowable observations for the PV= data set. (See the MAXOBS= option of the RESOURCE statement and CPM procedure in Chapter 4 for more details.)

NROUTCAL=*variable*

indicates which calendar is to be used for incrementing the _TIME_ variable in the BCOSTOUT= data set. (See the NROUTCAL= option of the RESOURCE statement and CPM procedure in Chapter 4 for more details.)

PV=*SAS-data-set*

identifies the output data set that contains the periodic planned value. The frequency is determined by the INTERVAL= and ROUTINTERVAL= parameters. The default data set name is PV.

RATE=*variable*

specifies the cost rate variable in the BUDGETCOST= data set. This is the rate at which cost is incurred over the duration of the task. The variable that contains this duration is specified with the DURATION= parameter. At least RATE= or COST= must be specified.

ROUTINTERVAL=*interval*

specifies the period between values in the PV= data set. (See the ROUTINTERVAL= option in Chapter 4 for more details.)

SPCT=*variable*

specifies the start percentage variable in the BUDGETCOST= data set. This is the percentage of the COST= amount that is incurred at the beginning of the task. The default is zero, in which case the entire COST= amount is incurred at the completion of the task. The specified variable may also contain a comma-delimited list of weights to be used for distributing the cost at equal working intervals over the duration of the activity.

TASKPV=*SAS-data-set*

identifies the output data set that is used as input to the %EVA_EARNED_VALUE macro. This data set is used to store the planned duration and costs by activity, which are used by the %EVA_EARNED_VALUE macro to compute earned value (see Figure 10.29). The default data set name is TASKPV.

WORKDAY=*SAS-data-set*

identifies the workday input data set. (See the WORKDAY= option in Chapter 4 for more details.)

Note that if a WORKDAY=, CALENDAR=, or HOLIDAY= data set is specified, the default variable name, _CAL_, is assumed for calendar identification.

Table 10.1 summarizes the parameters used for managing input and output data sets in the %EVA_PLANNED_VALUE macro.

Table 10.1 %EVA_PLANNED_VALUE I/O Parameter Summary

Type	Data Set	Variables
INPUT	BUDGETCOST	COST, RATE, SPCT
INPUT	CALENDAR	
INPUT	HOLIDATA	
INPUT	PLANSCHED	ACTIVITY, DURATION, FINISH, START
INPUT	WORKDAY	
OUTPUT	PV	
OUTPUT	TASKPV	

%EVA_EARNED_VALUE

This macro is used to produce the periodic earned value and actual cost.

%EVA_EARNED_VALUE *(parameters)* **;**

NOTE: %EVA_EARNED_VALUE requires output from %EVA_PLANNED_VALUE.

Required Parameters

ACTIVITY=*variable*

specifies the activity variable in the REVISESCHED=, ACTUALCOST=, and BUDGETCOST= data sets.

ACTUALCOST=*SAS-data-set*

REVISECOST=*SAS-data-set*

identifies the updated costs input data set.

REVISESCHED=*SAS-data-set*
identifies the input data set containing the updated schedule.

START=*variable*
specifies the start date or datetime variable in the REVISESCHED= data set.

Optional Parameters

CALENDAR=*SAS-data-set*
identifies the calendar input data set. (See the CALENDAR= option in Chapter 4 for more details.)

COST=*variable*
specifies the cost variable in the ACTUALCOST= data set. This is the total of the start and finish costs associated with the given task. At least COST= or RATE= must be specified. This parameter is only allowed if used for %EVA_PLANNED_VALUE; further, the variable specified must be the same as that specified for %EVA_PLANNED_VALUE. If a variable is specified for this parameter, but the value for the variable is missing for a given observation, the corresponding value from the BUDGETCOST= data set is used.

DURATION=*variable*
specifies the task duration variable in the REVISESCHED= data set. At least DURATION= or FINISH= must be specified. If both parameters are specified, the FINISH= variable value overrides a nonzero DURATION= variable value; i.e., duration is only used to indicate a milestone task.

EV=*SAS-data-set*
identifies the output data set, which contains the periodic earned value and revised cost. The frequency is determined by the INTERVAL= and ROUTINTERVAL= parameters. The default data set name is EV.

FINISH=*variable*
specifies the finish date or datetime variable in the REVISESCHED= data set. At least FINISH= or DURATION= must be specified. If both parameters are specified, the FINISH= variable value overrides a nonzero DURATION= variable value; i.e., duration is only used to indicate a milestone task.

HOLIDATA=*SAS-data-set*
identifies the input data set that contains the holiday schedule. (See the HOLIDATA= option in Chapter 4 for more details.)

HOLIDAY=*statement*
specifies the HOLIDAY statement to use for PROC CPM. For example:

```
%eva_earned_value(..., holiday=holiday start / holifin=finish, ...);
```

(See the HOLIDAY statement in Chapter 4 for more details.)

INTERVAL=*interval*
specifies the units to use for the DURATION= variable values. (See the INTERVAL= option in Chapter 4 for more details.)

MAXOBS=*max*

sets the limit on the number of allowable observations for the EV= data set. (See the MAXOBS= option of the RESOURCE statement and CPM procedure in Chapter 4 for more details.)

NROUTCAL=*variable*

indicates which calendar is to be used for incrementing the _TIME_ variable in the ACOSTOUT= data set. (See the NROUTCAL= option of the RESOURCE statement and CPM procedure in Chapter 4 for more details.)

RATE=*variable*

specifies the cost rate variable in the ACTUALCOST= data set. This is the rate at which cost is incurred over the duration of the task. The variable that contains this duration is specified with the DURATION= parameter. At least RATE= or COST= must be specified. This parameter is only allowed if used with %EVA_PLANNED_VALUE; further, the variable specified must be the same as that specified for %EVA_PLANNED_VALUE. If a variable is specified for this parameter, but the value for the variable is missing for a given observation, the corresponding value from the BUDGETCOST= data set is used.

ROUTINTERVAL=*interval*

specifies the period between values in the EV= data set. (See the ROUTINTERVAL= option in Chapter 4 for more details.)

SPCT=*variable*

specifies the start percentage variable in the ACTUALCOST= data set. This is the percentage of the COST= amount that is incurred at the beginning of the task. The default is zero, in which case the entire COST= amount is incurred at the completion of the task. The specified variable may also contain a comma-delimited list of weights to be used for distributing the cost at equal working intervals over the duration of the activity. This parameter is only allowed if used for %EVA_PLANNED_VALUE; further, the variable specified must be the same as that specified for %EVA_PLANNED_VALUE. If a variable is specified for this parameter, but the value for the variable is missing for a given observation, the corresponding value from the BUDGETCOST= data set will be used.

TASKPV=*SAS-data-set*

identifies the input data set containing the planned duration and costs by activity, which are used to compute earned value. This data set is generated by the %EVA_PLANNED_VALUE macro, with name specified by the TASKPV= parameter of the %EVA_PLANNED_VALUE macro (see Figure 10.29). The default data set name is TASKPV.

WORKDAY=*SAS-data-set*

identifies the workday input data set. (See the WORKDAY= option in Chapter 4 for more details.)

Note that if a WORKDAY=, CALENDAR=, or HOLIDAY= data set is specified, the default variable name, _CAL_, is assumed for calendar identification.

Also note that the variable names specified for the ACTIVITY=, START=, FINISH=, and RATE= parameters must be the same, respectively, as those previously specified for the %EVA_PLANNED_VALUE macro. This is to enable the subsequent use of the %EVA_TASK_METRICS and %EVG_GANTT_CHART macros.

Table 10.2 summarizes the parameters used for managing input and output data sets in the %EVA_EARNED_VALUE macro.

Table 10.2 %EVA_EARNED_VALUE I/O Parameter Summary

Type	Data Set	Variables
INPUT	ACTUALCOST	COST, RATE, SPCT
INPUT	CALENDAR	
INPUT	HOLIDATA	
INPUT	REVISESCHED	ACTIVITY, DURATION, FINISH, START
INPUT	TASKPV	COST, RATE, SPCT
INPUT	WORKDAY	
OUTPUT	EV	

%EVA_METRICS

This macro is used to generate summary statistics for the status date(s) requested, along with periodic metrics.

%EVA_METRICS < *(parameters)* > **;**

NOTE: %EVA_METRICS requires output from both %EVA_PLANNED_VALUE and %EVA_EARNED_VALUE.

Optional Parameters

ACRONYMS=*long*

specifies whether the long form of the earned value acronyms is to be used—e.g., BCWP versus EV. The short form is used unless the macro variable _ACRONYMS_ is set to "long" or ACRONYMS=long is specified. The parameter overrides the macro variable setting; i.e., specifying any non-null value other than "long" for the ACRONYMS= parameter produces short forms.

EACFORM=*integer*

indicates the form of the EAC to be used in computing ETC, VAC, VAC percentage, and TCPI_{EAC}. The values are assigned as follows:

- 1 - EAC_{rev}
- 2 - EAC_{OTD}
- 3 - EAC_{CPI}
- 4 - $\text{EAC}_{\text{CPI}\times\text{SPI}}$

The default value is 3 for EAC_{CPI}.

EV=*SAS-data-set*

identifies the input data set containing the periodic earned value and actual cost. It is generated by the %EVA_EARNED_VALUE macro, with name specified by the EV= parameter of the %EVA_EARNED_VALUE macro. The default data set name is EV.

METRICS=*SAS-data-set*

identifies the periodic output data set containing earned value metrics. The default name of the periodic output data set is METRICS.

PV=*SAS-data-set*

identifies the input data set containing the periodic planned value. It is generated by the %EVA_PLANNED_VALUE macro, with name specified by the PV= parameter of the %EVA_PLANNED_VALUE macro. The default data set name is PV.

SUMMARY=*SAS-data-set*

identifies the output data set, which contains a summary of metrics derived with respect to the last time given by the TIMENOW= parameter. The default name of the summary output data set is SUMMARY.

TIMENOW=*time(s)*

specifies one or more reference dates ('ddmmmyy'd) or datetimes ('ddmmmyy:hh:mm:ss'dt) delimited by white space, for the summary output. For example:

```
%eva_metrics(timenow='01SEP11'd '15SEP11:12:00:00'dt);
```

The list of dates or datetimes must be in chronological order. The default value is today's date, unless the macro variable _TIMENOW_ has been set, in which case the default is the latter.

Table 10.3 summarizes the parameters used for managing input and output data sets in the %EVA_METRICS macro.

Table 10.3 %EVA_METRICS I/O Parameter Summary

Type	Data Set
INPUT	EV
INPUT	PV
OUTPUT	METRICS
OUTPUT	SUMMARY

%EVA_TASK_METRICS

This macro is used to produce earned value measures for each task; these measures include Planned Value, Earned Value, Actual Cost, Cost and Schedule Variance, and Cost and Schedule Performance Index.

%EVA_TASK_METRICS *(parameters)* **;**

Required Parameters

ACTIVITY=*variable*

specifies the activity variable in the PLANSCHED= and REVISESCHED= data sets.

ACTUALCOST=*SAS-data-set*

REVISECOST=*SAS-data-set*

identifies the updated costs data set.

BUDGETCOST=*SAS-data-set*

PLANCOST=*SAS-data-set*

identifies the budgeted costs data set.

PLANSCHED=*SAS-data-set*

identifies the input data set that contains the planned schedule.

REVISESCHED=*SAS-data-set*

identifies the input data set that contains the updated schedule.

START=*variable*

specifies the start date or datetime variable in the PLANSCHED= and REVISESCHED= data sets.

Optional Parameters

ACRONYMS=*long*

specifies whether the long form of the earned value acronyms is to be used—e.g., BCWP versus EV. The short form is used unless the macro variable _ACRONYMS_ is set to "long" or ACRONYMS=long is specified. The parameter overrides the macro variable setting; i.e., specifying any non-null value other than "long" for the ACRONYMS= parameter produces short forms.

AGGREGATE=*Y or otherwise*

specifies whether or not to roll up values along the project hierarchy; a value of Y indicates that aggregation is to be carried out. No rollup is performed for the default behavior. The WBSCODE= parameter is required if AGGREGATE=Y is specified.

CALENDAR=*SAS-data-set*

identifies the calendar data set. (See the CALENDAR= option in Chapter 4 for more details.)

COST=*variable*

specifies the cost variable in the BUDGETCOST= and ACTUALCOST= data sets. This is the total of the start and finish costs associated with the given task. At least COST= or RATE= must be specified. If a variable is specified for this parameter, but the value for the variable is missing for a given observation, the corresponding value from the BUDGETCOST= data set is used.

DURATION=*variable*

specifies the task duration variable in the PLANSCHED= and REVISESCHED= data sets. At least DURATION= or FINISH= must be specified. If both parameters are specified, the FINISH= variable value overrides a nonzero DURATION= variable value; i.e., duration is only used to indicate a milestone task.

FINISH=*variable*

specifies the finish date or datetime variable in the PLANSCHED= and REVISESCHED= data sets At least FINISH= or DURATION= must be specified. If both parameters are specified, the FINISH= variable value overrides a nonzero DURATION= variable value; i.e., duration is only used to indicate a milestone task.

HOLIDATA=*SAS-data-set*

identifies the holiday data set. (See the HOLIDATA= option in Chapter 4 for more details.)

HOLIDAY=*statement*

specifies the HOLIDAY statement to use for PROC CPM. For example:

```
%eva_task_metrics(\ldots, holiday=holiday start / holifin=finish, ...);
```

(See the HOLIDAY statement in Chapter 4 for more details.)

INTERVAL=*interval*

specifies the units to use for the DURATION= variable values. (See the INTERVAL= option in Chapter 4 for more details.)

PCTCOMP=*variable*

specifies the percentage complete variable of the REVISESCHED= data set. Note that for a given activity, the specified percentage complete can appear to be in conflict with the amount of time spent, relative to the projected finish time for that activity. For example, even though 5 of 10 working days have elapsed, maybe only 30% of the activity has been completed. The input data reflects an anticipated acceleration in progress over the next five days. Also, if there is no value for the PCTCOMP= variable for a given activity, a computed value is substituted.

RATE=*variable*

specifies the cost rate variable in the BUDGETCOST= and ACTUALCOST= data sets. This is the rate at which cost is incurred over the DURATION= variable value for the given task. At least RATE= or COST= must be specified. If a variable is specified for this parameter, but the value for the variable is missing for a given observation, the corresponding value from the BUDGETCOST= data set is used.

SPCT=*variable*

specifies the start percentage variable in the BUDGETCOST= and ACTUALCOST= data sets. This is the percentage of the COST= amount that is incurred at the beginning of the task. The default is zero, in which case the entire COST= amount is incurred at the completion of the task. The specified variable may also contain a comma-delimited list of weights to be used for distributing the cost at equal working intervals over the duration of the activity. If a variable is specified for this parameter, but the value for the variable is missing for a given observation, the corresponding value from the BUDGETCOST= data set is used.

TASKMETRICS=*SAS-data-set*

identifies the output data set. The default data set name is TASKMETS.

TIMENOW=*time*

specifies the date ('ddmmmyy'd) or datetime ('ddmmmyy:hh:mm:ss'dt) of the updated schedule and costs. The default value is today's date unless the macro variable _TIMENOW_ has been set, in which case the default is the latter.

WBSCODE=*variable*

specifies the variable that contains the Work Breakdown Structure code in the PLANSCHED= or REVISESCHED= data set. The default variable name is WBS_CODE.

WORKDAY=*SAS-data-set*

identifies the workday input data set. (See the WORKDAY= option in Chapter 4 for more details.)

Note that if a WORKDAY=, CALENDAR=, or HOLIDAY= data set is specified, the default variable name, _CAL_, is assumed for calendar identification.

Table 10.4 summarizes the parameters used for managing input and output data sets in the %EVA_TASK_METRICS macro.

Table 10.4 %EVA_TASK_METRICS I/O Parameter Summary

Type	Data Set	Variables
INPUT	ACTUALCOST	COST, RATE, SPCT
INPUT	BUDGETCOST	COST, RATE, SPCT
INPUT	CALENDAR	
INPUT	HOLIDATA	
INPUT	PLANSCHED	ACTIVITY, DURATION, FINISH, START, WBSCODE
INPUT	REVISESCHED	ACTIVITY, DURATION, FINISH, PCTCOMP, START, WBSCODE
INPUT	WORKDAY	
OUTPUT	TASKMETRICS	

Charts

The following macros are available for earned value charts and reports.

%EVG_COST_PLOT < (*parameters*) > ;
%EVG_SCHEDULE_PLOT < (*parameters*) > ;
%EVG_INDEX_PLOT < (*parameters*) > ;
%EVG_VARIANCE_PLOT < (*parameters*) > ;
%EVG_GANTT_CHART (*parameters*) ;
%EVG_WBS_CHART (*parameters*) ;

%EVG_COST_PLOT

This macro is used to produce a plot of earned value measures over time. The plots include Planned Value, Earned Value, Actual Cost, and revised cost.

%EVG_COST_PLOT < (*parameters*) > ;

NOTE: %EVG_COST_PLOT requires output from %EVA_METRICS.

Optional Parameters

ACRONYMS=*long*

specifies whether the long form of the earned value acronyms is to be used—e.g., BCWP versus EV. The short form is used unless the macro variable _ACRONYMS_ is set to "long" or ACRONYMS=long is specified. The parameter overrides the macro variable setting; i.e., specifying any non-null value other than "long" for the ACRONYMS= parameter produces short forms.

METRICS=*SAS-data-set*

identifies the periodic metrics data set produced by the %EVA_METRICS macro, the name of which is determined by the METRICS= parameter of %EVA_METRICS. The default data set name is METRICS.

PLOT=*SAS-data-set*
specifies the name of the output data set. This is an internal data set that is used as input to the plotting procedure. The default data set name is PLOT.

SUMMARY=*SAS-data-set*
identifies the summary data set from the %EVA_METRICS macro, the name of which is determined by the SUMMARY= parameter of %EVA_METRICS. The default data set name is SUMMARY.

Table 10.5 summarizes the parameters used for managing input and output data sets in the %EVG_COST_PLOT macro.

Table 10.5 %EVG_COST_PLOT I/O Parameter Summary

Type	Data Set
INPUT	METRICS
INPUT	SUMMARY
OUTPUT	PLOT

%EVG_SCHEDULE_PLOT

This macro is used to display planned and projected completion times for the project. Specifically, the status date is shown, along with the planned finish time, the revised finish time, and the projected finish time.

%EVG_SCHEDULE_PLOT < *(parameters)* > **;**

NOTE: %EVG_SCHEDULE_PLOT requires output from %EVA_METRICS.

Optional Parameters

ACRONYMS=*long*
specifies whether the long form of the earned value acronyms is to be used—e.g., BCWP versus EV. The short form is used unless the macro variable _ACRONYMS_ is set to "long" or ACRONYMS=long is specified. The parameter overrides the macro variable setting; i.e., specifying any non-null value other than "long" for the ACRONYMS= parameter produces short forms.

METRICS=*SAS-data-set*
identifies the periodic metrics data set produced by the %EVA_METRICS macro, the name of which is determined by the METRICS= parameter of %EVA_METRICS. The default data set name is METRICS.

PLOT=*SAS-data-set*
specifies the name of the output data set. This is an internal data set that is used as input to the plotting procedure. The default data set name is PLOT.

Table 10.6 summarizes the parameters used for managing input and output data sets in the %EVG_SCHEDULE_PLOT macro.

Table 10.6 %EVG_SCHEDULE_PLOT I/O Parameter Summary

Type	Data Set
INPUT	METRICS
OUTPUT	PLOT

%EVG_INDEX_PLOT

This macro is used to obtain a plot of the cost and schedule performance indices over time. If the project has not been completed, the To-Complete Performance Index (Budget at Completion version) is also displayed. The legend includes the Planned Value, Earned Value, and Actual Cost.

%EVG_INDEX_PLOT *< (parameters) >* **;**

NOTE: %EVG_INDEX_PLOT requires output from %EVA_METRICS.

Optional Parameters

ACRONYMS=*long*
specifies whether the long form of the earned value acronyms is to be used—e.g., BCWP versus EV. The short form is used unless the macro variable _ACRONYMS_ is set to "long" or ACRONYMS=long is specified. The parameter overrides the macro variable setting; i.e., specifying any non-null value other than "long" for the ACRONYMS= parameter produces short forms.

METRICS=*SAS-data-set*
identifies the periodic metrics data set produced by the %EVA_METRICS macro, the name of which is determined by the METRICS= parameter of %EVA_METRICS. The default data set name is METRICS.

PLOT=*SAS-data-set*
specifies the name of the output data set. This is an internal data set that is used as input to the plotting procedure. The default data set name is PLOT.

SUMMARY=*SAS-data-set*
identifies the summary data set from the %EVA_METRICS macro, the name of which is determined by the SUMMARY= parameter of %EVA_METRICS. The default data set name is SUMMARY.

Table 10.7 summarizes the parameters used for managing input and output data sets in the %EVG_INDEX_PLOT macro.

Table 10.7 %EVG_INDEX_PLOT I/O Parameter Summary

Type	Data Set
INPUT	METRICS
INPUT	SUMMARY
OUTPUT	PLOT

%EVG_VARIANCE_PLOT

This macro is used to generate a plot of the cost and schedule variance against time; it features threshold plots obtained from the Planned Value, plus or minus 10%. The vital statistics of Planned Value, Earned Value, and Actual Cost are listed in the legend.

%EVG_VARIANCE_PLOT < *(parameters)* > **;**

NOTE: %EVG_VARIANCE_PLOT requires output from %EVA_METRICS.

Optional Parameters

ACRONYMS=*long*

specifies whether the long form of the earned value acronyms is to be used—e.g., BCWP versus EV. The short form is used unless the macro variable _ACRONYMS_ is set to "long" or ACRONYMS=long is specified. The parameter overrides the macro variable setting; i.e., specifying any non-null value other than "long" for the ACRONYMS= parameter produces short forms.

METRICS=*SAS-data-set*

identifies the periodic metrics data set produced by the %EVA_METRICS macro, the name of which is determined by the METRICS= parameter of %EVA_METRICS. The default data set name is METRICS.

PLOT=*SAS-data-set*

specifies the name of the output data set. This is an internal data set that is used as input to the plotting procedure.

The default data set name is PLOT.

Table 10.8 summarizes the parameters used for managing input and output data sets in the %EVG_VARIANCE_PLOT macro.

Table 10.8 %EVG_VARIANCE_PLOT I/O Parameter Summary

Type	Data Set
INPUT	METRICS
OUTPUT	PLOT

%EVG_GANTT_CHART

This macro is used to produce a Gantt chart of the planned and revised schedule, along with selected earned value metrics for each task.

%EVG_GANTT_CHART *(parameters)* **;**

NOTE: %EVG_GANTT_CHART requires output from %EVA_METRICS.

Required Parameters

ACTIVITY=*variable*
specifies the activity variable in the PLANSCHED= and REVISESCHED= data sets.

FINISH=*variable*
specifies the finish date or datetime variable in the PLANSCHED= and REVISESCHED= data sets.

PLANSCHED=*SAS-data-set*
identifies the data set containing the planned schedule.

REVISESCHED=*SAS-data-set*
identifies the data set that contains the updated schedule.

START=*variable*
specifies the start date or datetime variable in the PLANSCHED= and REVISESCHED= data sets.

Optional Parameters

CHART=*SAS-data-set*
specifies the name of the output data set. This is an internal data set that is used as input to the GANTT procedure. The default data set name is CHART.

DURATION=*variable*
specifies a duration variable in the PLANSCHED= or REVISESCHED= data set. The specified variable in the REVISESCHED= data set is used if the variable is present in both data sets. This variable is used only for differentiating milestones from single-day tasks.

HEIGHT=*h*
specifies a height factor for all text in PROC GANTT. (See the HEIGHT= option for the CHART statement of PROC GANTT (Chapter 8) for more details.)

HPAGES=*h*
specifies that the Gantt chart is to be produced using *h* horizontal pages. (See the HPAGES= option for the CHART statement of PROC GANTT (Chapter 8) for more details.)

VPAGES=*v*
specifies that the Gantt chart is to be produced using *v* vertical pages. (See the VPAGES= option for the CHART statement of PROC GANTT (Chapter 8) for more details.)

ID=*variable(s)*
specifies a space-delimited list of the variables from the TASKS= data set to be included in the chart. The choices are: WBS, PV, EV, AC, CV, CVP, SV, SVP, CPI, and SPI. The default value is “CV SV”.

SCALE=*scale*
specifies the relative size adjustment of the data columns and chart. (See the SCALE= option for the CHART statement of PROC GANTT (Chapter 8) for more details.)

TASKMETRICS=*SAS-data-set*
specifies the name of the output data set from the %EVA_TASK_METRICS macro. The default data set name is TASKMETS.

TIMENOW=*time*
specifies the date ('ddmmmyy'd) or datetime ('ddmmmyy:hh:mm:ss'dt) of the updated schedule and costs. The default value is today's date, unless the macro variable _TIMENOW_ has been set, in which case the default is the latter. A vertical line and label indicate the value.

Table 10.9 summarizes the parameters used for managing input and output data sets in the %EVG_GANTT_CHART macro.

Table 10.9 %EVG_GANTT_CHART I/O Parameter Summary

Type	Data Set	Variables
INPUT	PLANSCHED	ACTIVITY, DURATION, FINISH, START
INPUT	REVISESCHED	ACTIVITY, DURATION, FINISH, START
INPUT	TASKMETRICS	
OUTPUT	CHART	

%EVG_WBS_CHART

This macro is used to generate a Work Breakdown Structure chart. The parameters can be used to control the data that appears within the boxes of the display.

%EVG_WBS_CHART *(parameters)* **;**

Required Parameters

ACTIVITY=*variable*
specifies the activity variable in the STRUCTURE= data set.

PROJECT=*variable*
specifies the STRUCTURE= data set variable that identifies the project to which the current task belongs.

STRUCTURE=*SAS-data-set*
identifies the data set that defines the Work Breakdown Structure.

Optional Parameters

DEFID=*Y or otherwise*
specifies whether or not to include the default identification variables that are present in the STRUCTURE= data set in the boxes of the display. Default variable names are: E_START, E_FINISH, L_START, L_FINISH, S_START, S_FINISH, A_START, A_FINISH, T_FLOAT, and F_FLOAT. The default value for the DEFID= parameter is Y, meaning the default variables are displayed. (See the NODEFID option of the ACTNET statement of the NETDRAW procedure (Chapter 9) for more details.)

ID=*variable list*
specifies the STRUCTURE= data set variables that appear in the boxes of the display.

REVERSEY=*Y or otherwise*

specifies whether or not to reverse the order in which the output pages appear. The default value is Y, which orders the pages from top to bottom. (See the REVERSEY option of the ACTNET statement of the NETDRAW procedure (Chapter 9) for more details.)

ROTATE=*Y or otherwise*

specifies whether or not to rotate the display by 90 degrees. The default value is Y, which performs the rotation. (See the ROTATE option of the ACTNET statement of the NETDRAW procedure (Chapter 9) for more details.)

ROTATETEXT=*Y or otherwise*

specifies whether or not to rotate the text that appears within the boxes of the display by 90 degrees. The default value is Y, which rotates the text. (See the ROTATETEXT option of the ACTNET statement of the NETDRAW procedure (Chapter 9) for more details.)

VPAGES=*integer*

specifies the number of vertical pages to produce. The default value is 1. (See the VPAGES= parameter of the ACTNET statement of the NETDRAW procedure (Chapter 9) for more details.)

Table 10.10 summarizes the parameters used for managing input and output data sets in the %EVG_WBS_CHART macro.

Table 10.10 %EVG_WBS_CHART I/O Parameter Summary

Type	Data Set	Variables
INPUT	STRUCTURE	ACTIVITY, ID, PROJECT

Details: Earned Value Management Macros

Variances

Table 10.11 shows the formulas for the Cost Variance, Schedule Variance, and Variance at Completion, along with the associated percentages.

Table 10.11 Variance Formulas

Acronym	Formula
CV	EV - AC
CV%	$\frac{\text{CV}}{\text{EV}}$
SV	EV - PV
SV%	$\frac{\text{SV}}{\text{PV}}$
VAC	BAC - EAC
VAC%	$\frac{\text{VAC}}{\text{BAC}}$

Performance Indices

Table 10.12 shows the formulas for Cost, Schedule, and To-Complete Performance Indices. WR is Work Remaining and is given by: $\mathrm{WR} = \mathrm{BAC} - \mathrm{EV}$. The section "Cost Estimates" on page 826 gives different expressions for the EAC (Estimate at Completion).

Table 10.12 Performance Index Formulas

Acronym	Formula
CPI	$\frac{\mathrm{EV}}{\mathrm{AC}}$
SPI	$\frac{\mathrm{EV}}{\mathrm{PV}}$
$\mathrm{TCPI}_{\mathrm{BAC}}$	$\frac{\mathrm{WR}}{\mathrm{BAC}-\mathrm{AC}}$
$\mathrm{TCPI}_{\mathrm{EAC}}$	$\frac{\mathrm{WR}}{\mathrm{EAC}-\mathrm{AC}}$

Cost Estimates

The ETC, or Estimate to Complete, and the EAC, or Estimate at Completion, are two metrics that are central to Earned Value Analysis. The ETC is a projection of the additional funds needed to finish the project. The EAC can be derived as follows (where AC represents the actual costs to date):

$EAC = AC + ETC.$

For the purposes of this document, each of these two cost estimates takes four forms, illustrated by the following table for ETC:

Table 10.13 ETC Formulas

Acronym	Description	Formula
$\mathrm{ETC}_{\mathrm{rev}}$	Revised	$\mathrm{EAC}_{\mathrm{rev}}$ - AC
$\mathrm{ETC}_{\mathrm{OTD}}$	Overrun to date	BAC - EV = WR
$\mathrm{ETC}_{\mathrm{CPI}}$	CPI	$\frac{\mathrm{WR}}{\mathrm{CPI}}$
$\mathrm{ETC}_{\mathrm{CPI}\times\mathrm{SPI}}$	CPI times SPI	$\frac{\mathrm{WR}}{\mathrm{CPI}\times\mathrm{SPI}}$

The $\mathrm{ETC}_{\mathrm{rev}}$ is derived directly from the revised schedule, revised costs, and actual costs to date. This form does not make allowances for past performance. It is assumed that this information has already been factored into the updated schedule and costs. Analogous to the $\mathrm{ETC}_{\mathrm{rev}}$, the $\mathrm{ETC}_{\mathrm{OTD}}$ is taken to be the remaining planned value (Budget at Completion less Earned Value), also known as Work Remaining (WR). This form can be thought of as having a performance factor of 1.

The $\mathrm{ETC}_{\mathrm{CPI}}$ form is the remaining planned value divided by the CPI. In this way favorable cost performance (CPI greater than 1) forces the estimate downward. The performance factor in this case is the inverse of the CPI. Similarly, the $\mathrm{ETC}_{\mathrm{CPI}\times\mathrm{SPI}}$ form employs the inverse of the product of the CPI and SPI as the performance factor. In this case, both cost and schedule performance affect the estimate. For example, if both CPI and SPI are favorable (greater than 1), the estimate is lower.

Analysis

This section describes each macro in the earned value macro set in further detail.

%EVA_PLANNED_VALUE

This macro is used to establish periodic planned value. The first input data set for this macro (identified using the PLANSCHED= parameter) provides an initial baseline schedule. The second input data set (identified using the BUDGETCOST= parameter) contains the budgeted costs data set. One of the output data sets (identified using the PV= parameter) consists of two pertinent variables: _TIME_ and _PV_RATE_. The latter variable lists the budgeted cost for the date or datetime indicated by the former variable. Another output data set is identified using the TASKPV= parameter, and provides the planned duration and costs by activity, for later use by the %EVA_EARNED_VALUE macro.

Upon completion, the status of the macro is saved in a global macro variable named _EVA_PLANNED_VALUE_, which can take one of the following three values:

- STATUS=SUCCESSFUL indicates successful completion of the macro.
- STATUS=SYNTAX_ERROR indicates that macro execution was halted due to an error in the macro invocation syntax.
- STATUS=RUNTIME_ERROR indicates that the macro invocation failed.

%EVA_EARNED_VALUE

This macro is used to establish periodic earned value and actual cost. The first of the input data sets for this macro (identified using the REVISESCHED= parameter) contains a schedule that has been updated to reflect the current status of the project. A second input data set (identified using the ACTUALCOST= parameter) specifies the revised cost. The third input data set, from the %EVA_PLANNED_VALUE macro, is identified using the TASKPV= parameter, and supplies the planned duration and costs by activity. These are needed for computing the earned value. The output data set (identified using the EV= parameter) contains three relevant variables: _TIME_, _EV_RATE_, and _AC_RATE_. _EV_RATE_ lists the Earned Value, and _AC_RATE_ the Actual Cost of Work Performed, for the date or datetime indicated by the _TIME_ variable.

Upon completion, the status of the macro is saved in a global macro variable named _EVA_EARNED_VALUE_, which can take one of the following three values:

- STATUS=SUCCESSFUL indicates successful completion of the macro.
- STATUS=SYNTAX_ERROR indicates that macro execution was halted due to an error in the macro invocation syntax.
- STATUS=RUNTIME_ERROR indicates that the macro invocation failed.

%EVA_METRICS

This macro is used to compute the periodic metrics and summary statistics in reference to the date(s) or datetime(s) specified by the TIMENOW= parameter. The periodic planned value data set (generated by the %EVA_PLANNED_VALUE macro) and the periodic earned value data set (generated by the %EVA_EARNED_VALUE macro) are the inputs to the %EVA_METRICS macro. These two data sets are identified using the PV= and EV= parameters, respectively. The METRICS= parameter identifies the periodic output data set, and the SUMMARY= parameter identifies the summary output data set. An output listing is also produced, which is derived from the summary output data set. The two output data sets are described next.

Periodic Output Data Set

The variables in this data set are as follows:

- _TIME_ is the date or datetime reference point for the other variables.
- _PV_ is the cumulative Planned Value (Budgeted Cost of Work Scheduled).
- _EV_ is the cumulative Earned Value (Budgeted Cost of Work Performed).
- _AC_ is the cumulative Actual Cost (Actual Cost of Work Performed).
- _EACREV_ is the $\mathrm{EAC}_{\mathrm{rev}}$. This variable is simply an accumulation of costs to project completion, according to the revised budget and schedule.
- _SV_ is the cumulative Schedule Variance, calculated as $SV = EV - PV$.
- _CV_ is the cumulative Cost Variance, calculated as $CV = EV - AC$.
- _CPI_ is the Cost Performance Index, calculated as $CPI = \frac{EV}{AC}$.
- _SPI_ is the Schedule Performance Index, calculated as $SPI = \frac{EV}{PV}$.

Summary Output Data Set

The variables in this data set are as follows:

- _NAME_ is the internal variable name.
- _LABEL_ is the label used for the variable in the output listing.
- _VALUE*n*_ gives the value of the variable for the reference date or datetime given by the respective TIMENOW= parameter value.

Each row of the data set corresponds to one of the summary statistics as follows:

- _pctcomp_ is the percentage complete for the entire project, calculated as $Percentage\ Complete = \frac{EV}{BAC}$.
- _ev_ is the cumulative Budgeted Cost of Work Performed (Earned Value).

- _pv_ is the cumulative Budgeted Cost of Work Scheduled (Planned Value).
- _ac_ is the cumulative Actual Cost of Work Performed (Actual Cost).
- _cpi_ is the Cost Performance Index.
- _spi_ is the Schedule Performance Index.
- _cv_ is the Cost Variance, calculated as $CV = EV - AC$.
- _cvp_ is the Cost Variance percentage, calculated as $CV\% = \frac{CV}{EV}$.
- _sv_ is the Schedule Variance, calculated as $SV = EV - PV$.
- _svp_ is the Schedule Variance percentage, calculated as $SV\% = \frac{SV}{PV}$.
- _bac_ is the Budget at Completion (BAC).
- _etc_ is the Estimate To Complete (ETC). The form of EAC used to compute the ETC is specified by the EACFORM= parameter of the %EVA_METRICS macro.
- _eacrev_ is the revised Estimate at Completion ($\mathrm{EAC_{rev}}$).
- _eacotd_ is the Overrun to Date form of the EAC, calculated as $EAC_{OTD} = AC + WR$, where WR is the Work Remaining ($WR = BAC - EV$).
- _eaccpi_ is the Cumulative CPI form of the EAC, calculated as $EAC_{CPI} = AC + \frac{WR}{CPI}$.
- _eaccpispi_ is the Cumulative CPI times SPI form of the EAC, calculated as $EAC_{CPI \times SPI} = AC + \frac{WR}{CPI \times SPI}$.
- _vac_ is the Variance at Completion, calculated as $VAC = BAC - EAC$. The form of EAC used is specified by the EACFORM= parameter of the %EVA_METRICS macro.
- _vacp_ is the CPI Variance at Completion percentage, calculated as $VAC\% = \frac{VAC}{BAC}$. The form of EAC used to compute the VAC is specified by the EACFORM= parameter of the %EVA_METRICS macro.
- _tcpi_ is the BAC To-Complete Performance Index, calculated as $TCPI = \frac{WR}{BAC - AC}$.
- _etcpi_ is the EAC To-Complete Performance Index, calculated as $TCPI = \frac{WR}{EAC - AC}$. The form of EAC used is specified by the EACFORM= parameter of the %EVA_METRICS macro.

Upon completion, the status of the macro is saved in a global macro variable named _EVA_METRICS_, which can take one of the following three values:

- STATUS=SUCCESSFUL indicates successful completion of the macro.
- STATUS=SYNTAX_ERROR indicates that macro execution was halted due to an error in the macro invocation syntax.
- STATUS=RUNTIME_ERROR indicates that the macro invocation failed.

%EVA_TASK_METRICS

This macro uses the budgeted schedule and costs (identified using the PLANSCHED= and BUDGETCOST= parameters, respectively), along with the updated schedule and costs (identified using the REVISESCHED= and ACTUALCOST= parameters, respectively), to produce Cost Variance and Schedule Variance by task. The corresponding output data set is specified using the TASKMETRICS= parameter and is represented in the output listing. The variables in this data set are as follows:

- Activity is the reference task for the given row.
- WBS_CODE is the Work Breakdown Structure code for the given task.
- _CV_ is the Cost Variance for the given task.
- _CVP_ is the Cost Variance percentage for the given task.
- _SV_ is the Schedule Variance for the given task.
- _SVP_ is the Schedule Variance percentage for the given task.
- _CPI_ is the Cost Performance Index for the given task.
- _SPI_ is the Schedule Performance Index for the given task.
- _PV_ is the Planned Value (BCWS) for the given task.
- _EV_ is the Earned Value (BCWP) for the given task.
- _AC_ is the Actual Cost (ACWP) for the given task.

Specifying “AGGREGATE=Y” with the WBSCODE= option forces the preceding metrics to be rolled up along the project hierarchy.

Upon completion, the status of the macro is saved in a global macro variable named _EVA_TASK_METRICS_, which can take one of the following three values:

- STATUS=SUCCESSFUL indicates successful completion of the macro.
- STATUS=SYNTAX_ERROR indicates that macro execution was halted due to an error in the macro invocation syntax.
- STATUS=RUNTIME_ERROR indicates that the macro invocation failed.

%EVG_COST_PLOT

This macro is used to generate a line plot of the Earned Value, Planned Value, Actual Cost, and revised cost over time. If the project has not been completed, three Estimates at Completion are also delineated by a three-tiered box at the right-hand side of the plot area. In addition, several vital statistics appear as part of the legend: Planned Value, Earned Value, Actual Cost, revised cost, Cost and Schedule Performance Indices, Budget At Completion, and Estimates at Completion.

Upon completion, the status of the macro is saved in a global macro variable named _EVG_COST_PLOT_, which can take one of the following three values:

- STATUS=SUCCESSFUL indicates successful completion of the macro.
- STATUS=SYNTAX_ERROR indicates that macro execution was halted due to an error in the macro invocation syntax.
- STATUS=RUNTIME_ERROR indicates that the macro invocation failed.

%EVG_SCHEDULE_PLOT

This macro is used to visualize planned and projected completion times for the project. To this end, plots of the planned value and actual and revised cost are displayed. To arrive at a projected finish time, the earned value plot is extended to the Budget at Completion with a slope equal to the ratio of Earned Value to time elapsed at the status date. The slope of this extension is the average rate at which Earned Value has accumulated up to the status date. Vital statistics displayed in the legend include Planned Value, Earned Value, Actual Cost, and Budget at Completion.

Upon completion, the status of the macro is saved in a global macro variable named _EVG_SCHEDULE_PLOT_, which can take one of the following three values:

- STATUS=SUCCESSFUL indicates successful completion of the macro.
- STATUS=SYNTAX_ERROR indicates that macro execution was halted due to an error in the macro invocation syntax.
- STATUS=RUNTIME_ERROR indicates that the macro invocation failed.

%EVG_INDEX_PLOT

This macro is used to plot the Cost and Schedule Performance Indices over time. The To-Complete Performance Index is also shown.

Upon completion, the status of the macro is saved in a global macro variable named _EVG_INDEX_PLOT_, which can take one of the following three values:

- STATUS=SUCCESSFUL indicates successful completion of the macro.
- STATUS=SYNTAX_ERROR indicates that macro execution was halted due to an error in the macro invocation syntax.
- STATUS=RUNTIME_ERROR indicates that the macro invocation failed.

%EVG_VARIANCE_PLOT

This macro is used to plot the Cost and Schedule Variances over time. Upon completion, the status of the macro is saved in a global macro variable named _EVG_VARIANCE_PLOT_, which can take one of the following three values:

- STATUS=SUCCESSFUL indicates successful completion of the macro.

- STATUS=SYNTAX_ERROR indicates that macro execution was halted due to an error in the macro invocation syntax.
- STATUS=RUNTIME_ERROR indicates that the macro invocation failed.

%EVG_GANTT_CHART

This macro is used to show the Cost and Schedule Variances and the Cost and Schedule Performance Indices by activity, along with a Gantt chart of the schedule. For each task, the bottom bar depicts the baseline schedule and the top bar represents the updated schedule.

Upon completion, the status of the macro is saved in a global macro variable named _EVG_GANTT_CHART_, which can take one of the following three values:

- STATUS=SUCCESSFUL indicates successful completion of the macro.
- STATUS=SYNTAX_ERROR indicates that macro execution was halted due to an error in the macro invocation syntax.
- STATUS=RUNTIME_ERROR indicates that the macro invocation failed.

%EVG_WBS_CHART

This macro is used to display the Work Breakdown Structure, in addition to any other data associated with an activity.

Upon completion, the status of the macro is saved in a global macro variable named _EVG_WBS_CHART_, which can take one of the following three values:

- STATUS=SUCCESSFUL indicates successful completion of the macro.
- STATUS=SYNTAX_ERROR indicates that macro execution was halted due to an error in the macro invocation syntax.
- STATUS=RUNTIME_ERROR indicates that the macro invocation failed.

Examples: Earned Value Management Macros

In this section, a series of examples illustrates the usage and results of the earned value macro set.

Example 10.1: Integrated Assembly Project

The planned schedule for an assembly project is shown in Output 10.1.1. This schedule can be computed by either of the CPM or PM procedures. (For more details, see Chapter 4 or Chapter 5, respectively.)

Output 10.1.1 Schedule IOUT1

Integrated Assembly Test Project
Initial Schedule

Obs	Activity	Duration	Description	Planned Start	Planned Finish
1	S	0	Start	01OCT09	01OCT09
2	PD	105	Preliminary Design	01OCT09	13JAN10
3	PDR	21	Prelim Design Review	14JAN10	03FEB10
4	FD	168	Final Design	04FEB10	21JUL10
5	PM	126	Procure Material	10JUN10	13OCT10
6	FDR	21	Final Design Review	22JUL10	11AUG10
7	FP	273	Facility Preparation	01OCT10	30JUN11
8	FC	273	Fabricate Components	12AUG10	11MAY11
9	DA	26	Deliver Assembly	26JUN11	21JUL11
10	FRR	21	Facil Readiness Rvw	01JUL11	21JUL11
11	IA	42	Install Assembly	22JUL11	01SEP11
12	RR	21	Readiness Review	02SEP11	22SEP11
13	T	126	Test	01OCT11	03FEB12
14	TV	63	Test Validation	07JAN12	09MAR12

The budgeted cost rates are given in Output 10.1.2. It is assumed that the activities incur costs continuously.

Output 10.1.2 Cost Rates IATCOST

Budgeted Costs

Obs	Activity	Rate
1	S	0.0000
2	PD	4.7619
3	PDR	7.1429
4	FD	1.7857
5	PM	6.0317
6	FDR	7.1429
7	FP	4.0293
8	FC	5.1282
9	DA	13.0769
10	FRR	7.1429
11	IA	5.9524
12	RR	7.1429
13	T	7.1429
14	TV	7.9365

The budgeted periodic cost can now be generated with the following specification of the %EVA_PLANNED_VALUE macro:

```
%eva_planned_value(
    plansched=iout1,
    activity=activity,
    start=start,
    finish=finish,
    duration=duration,
    budgetcost=iatcost,
    rate=rate
);
```

For brevity, only the first 17 rows of the output data set are shown in Output 10.1.3.

Output 10.1.3 %EVA_PLANNED_VALUE: Periodic Data Set

Daily Planned Value

Obs	Period Identifier	PV Rate
1	01OCT09	4.76190
2	02OCT09	4.76190
3	03OCT09	4.76190
4	04OCT09	4.76190
5	05OCT09	4.76190
6	06OCT09	4.76190
7	07OCT09	4.76190
8	08OCT09	4.76190
9	09OCT09	4.76190
10	10OCT09	4.76190
11	11OCT09	4.76190
12	12OCT09	4.76190
13	13OCT09	4.76190
14	14OCT09	4.76190
15	15OCT09	4.76190
16	16OCT09	4.76190
17	17OCT09	4.76190

Next, the actual progress of the project through September 30, 2010, is entered. The ACTUAL data set is shown in Output 10.1.4.

Output 10.1.4 Current Status ACTUAL

Status

Obs	Activity	Actual Start	Actual Finish	Actual Rate	Pct. Comp.
1	S	01OCT09	01OCT09	0.0	100
2	PD	01OCT09	29JAN10	6.0	100
3	PDR	30JAN10	19FEB10	8.2	100
4	FD	20FEB10	10SEP10	3.1	100
5	PM	10AUG10	.	6.4	40
6	FDR	11SEP10	.	8.0	80

These inputs are then used by the CPM procedure with the original schedule to produce an updated schedule, given in Output 10.1.5. (For information about using the CPM procedure, see Chapter 4.)

Output 10.1.5 Updated Schedule UPDSCHED

Updated Schedule

Obs	Activity	Actual Duration	Description	Pct. Comp.	Start	Finish
1	S	0	Start	100	01OCT09	01OCT09
2	PD	121	Preliminary Design	100	01OCT09	29JAN10
3	PDR	21	Prelim Design Review	100	30JAN10	19FEB10
4	FD	203	Final Design	100	20FEB10	10SEP10
5	PM	.	Procure Material	40	10AUG10	17DEC10
6	FDR	.	Final Design Review	80	11SEP10	05OCT10
7	FP	.	Facility Preparation	.	06OCT10	05JUL11
8	FC	.	Fabricate Components	.	12OCT10	11JUL11
9	DA	.	Deliver Assembly	.	07JUL11	01AUG11
10	FRR	.	Facil Readiness Rvw	.	06JUL11	26JUL11
11	IA	.	Install Assembly	.	02AUG11	12SEP11
12	RR	.	Readiness Review	.	13SEP11	03OCT11
13	T	.	Test	.	04OCT11	06FEB12
14	TV	.	Test Validation	.	10JAN12	12MAR12

The %EVA_EARNED_VALUE macro can then be used to generate the updated periodic cost, as follows:

```
%eva_earned_value(
    revisesched=updsched,
    activity=activity,
    start=start,
    finish=finish,
    actualcost=iatupd,
    rate=rate
);
```

Again, for brevity only the first 17 rows of the output data set are shown in Output 10.1.6.

Output 10.1.6 %EVA_EARNED_VALUE: Periodic Data Set

Daily Earned Value and Revised Cost

Obs	Period Identifier	EV Rate	AC Rate
1	01OCT09	4.13223	6
2	02OCT09	4.13223	6
3	03OCT09	4.13223	6
4	04OCT09	4.13223	6
5	05OCT09	4.13223	6
6	06OCT09	4.13223	6
7	07OCT09	4.13223	6
8	08OCT09	4.13223	6
9	09OCT09	4.13223	6
10	10OCT09	4.13223	6
11	11OCT09	4.13223	6
12	12OCT09	4.13223	6
13	13OCT09	4.13223	6
14	14OCT09	4.13223	6
15	15OCT09	4.13223	6
16	16OCT09	4.13223	6
17	17OCT09	4.13223	6

The %EVA_METRICS macro can be called with a current date of September 30, 2010, as follows:

```
%eva_metrics(
    timenow='30sep10'd,
    acronyms=long
);
```

Output 10.1.7 shows the output listing from %EVA_METRICS. Notice that "ACRONYMS=long" is specified, which results in the long version of the earned value acronyms being used.

Output 10.1.7 %EVA_METRICS: Summary Statistics

Earned Value Analysis
as of September 30, 2010

Metric	Value
Percent Complete	20.66
BCWS (Budgeted Cost of Work Scheduled)	2038.00
BCWP (Budgeted Cost of Work Performed)	1374.00
ACWP (Actual Cost of Work Performed)	2020.30
CV (Cost Variance)	-646.30
CV%	-47.04
SV (Schedule Variance)	-664.00
SV%	-32.58
CPI (Cost Performance Index)	0.68
SPI (Schedule Performance Index)	0.67
BAC (Budget At Completion)	6650.00
EAC (Revised Estimate At Completion)	7349.50
EAC (Overrun to Date)	7296.30
EAC (Cumulative CPI)	9778.02
EAC (Cumulative CPI X SPI)	13527.00
ETC (Estimate To Complete)*	7757.72
VAC (Variance At Completion)*	-3128.02
VAC%*	-47.04
TCPI (BAC) (To-Complete Performance Index)	1.14
TCPI (EAC) (To-Complete Performance Index)*	0.68

*** The CPI form of the EAC is used.**

Next, the %EVA_TASK_METRICS macro is used to produce Cost and Schedule Variance by task.

```
%eva_task_metrics(
    plansched=iout1,
    revisesched=updsched,
    activity=activity,
    start=start,
    finish=finish,
    pctcomp=pctcomp,
    budgetcost=iatcost,
    actualcost=iatupd,
    rate=rate,
    timenow='30sep10'd,
    acronyms=long
);
```

The output listing is shown in Output 10.1.8.

Output 10.1.8 %EVA_TASK_METRICS: CV and SV by Activity

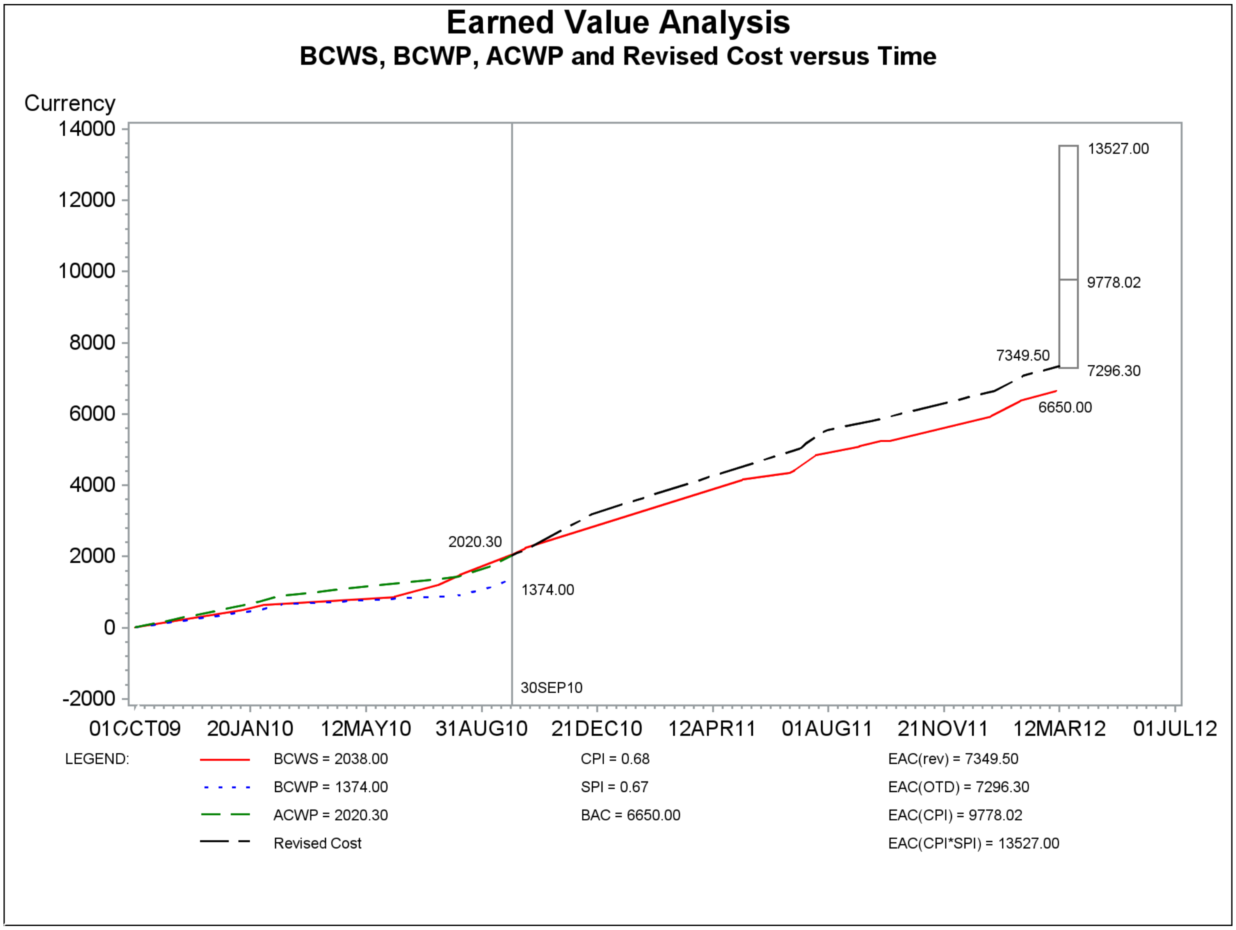
Earned Value Analysis by Activity
as of September 30, 2010

Obs	Activity	BCWS	BCWP	ACWP	CV	CV%	SV	SV%	CPI	SPI
1	S	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.	.
2	PD	500.00	500.00	726.00	-226.00	-45.20	0.00	0.00	0.69	1.00
3	PDR	150.00	150.00	172.20	-22.20	-14.80	0.00	0.00	0.87	1.00
4	FD	300.00	300.00	629.30	-329.30	-109.77	0.00	0.00	0.48	1.00
5	PM	681.59	304.00	332.80	-28.80	-9.47	-377.59	-55.40	0.91	0.45
6	FDR	150.00	120.00	160.00	-40.00	-33.33	-30.00	-20.00	0.75	0.80
7	FP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.	.
8	FC	256.41	0.00	0.00	0.00	0.00	-256.41	-100.00	.	0.00
9	DA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.	.
10	FRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.	.
11	IA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.	.
12	RR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.	.
13	T	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.	.
14	TV	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.	.

Output 10.1.9 through Output 10.1.13 show charts that are produced using the earned value analysis reporting macros. First, the %EVG_COST_PLOT macro is used to generate the plot in Output 10.1.9.

```
%evg_cost_plot(acronyms=long);
```

Output 10.1.9 %EVG_COST_PLOT, EV, PV, AC, and EAC_{rev}

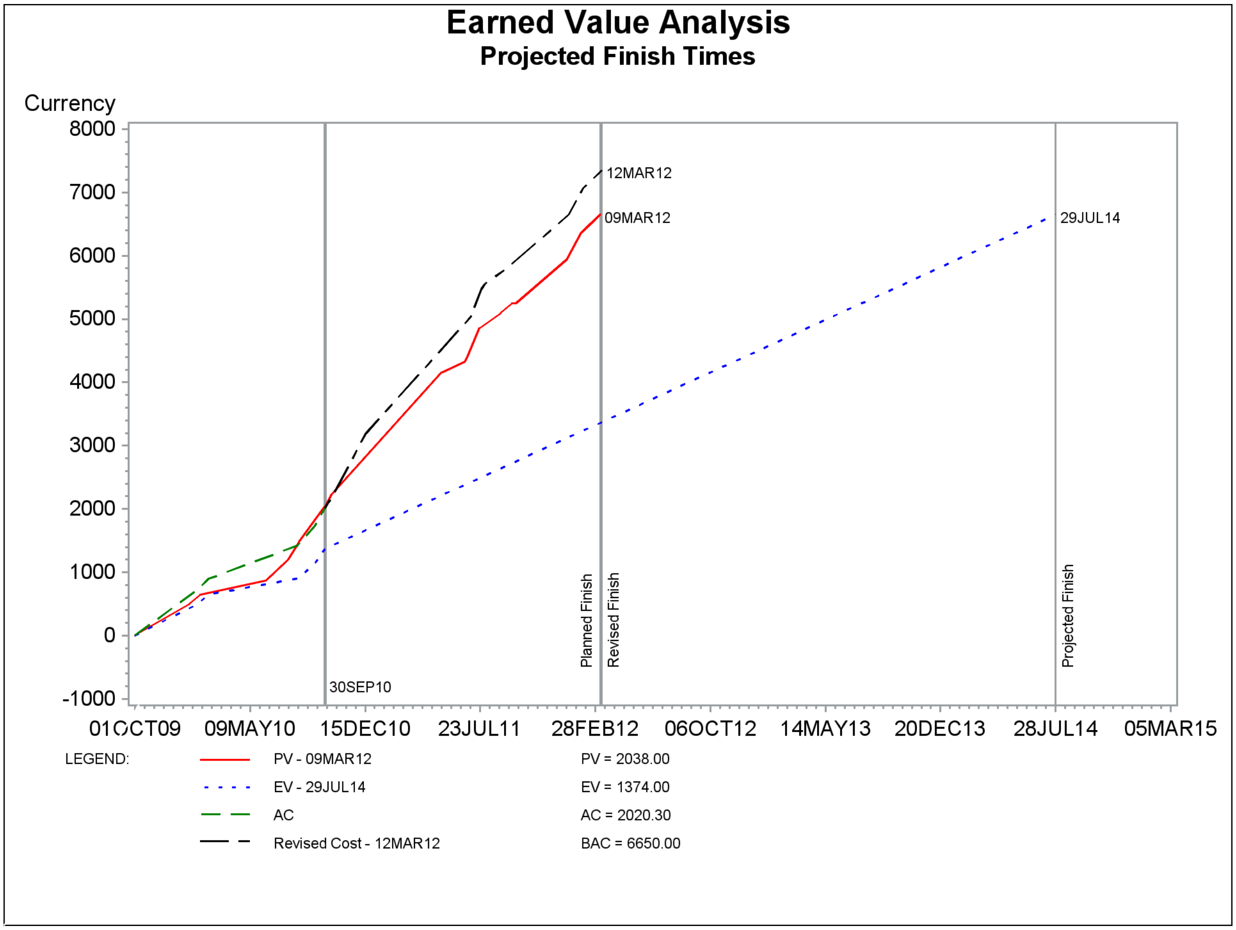


According to the plan, the earned value percentage complete at the status date of September 30, 2010 (shown by the BCWS plot) should have been 2049/6650, or 30.81%. Instead, the percentage complete (shown by the BCWP plot) is only 1374/6650, or 20.66%.

Next, the %EVG_SCHEDULE_PLOT macro is used to produce the plot in Output 10.1.10. The resulting output shows a disastrous projected completion date of August 1, 2014, based upon the current earned value. This is two years and four months behind the planned schedule end date of March 10, 2012. Based on the performance of the project so far, it is estimated to cost $9793 at completion ($EAC_{CPI}$), amounting to nearly a 50% overrun.

```
%evg_schedule_plot;
```

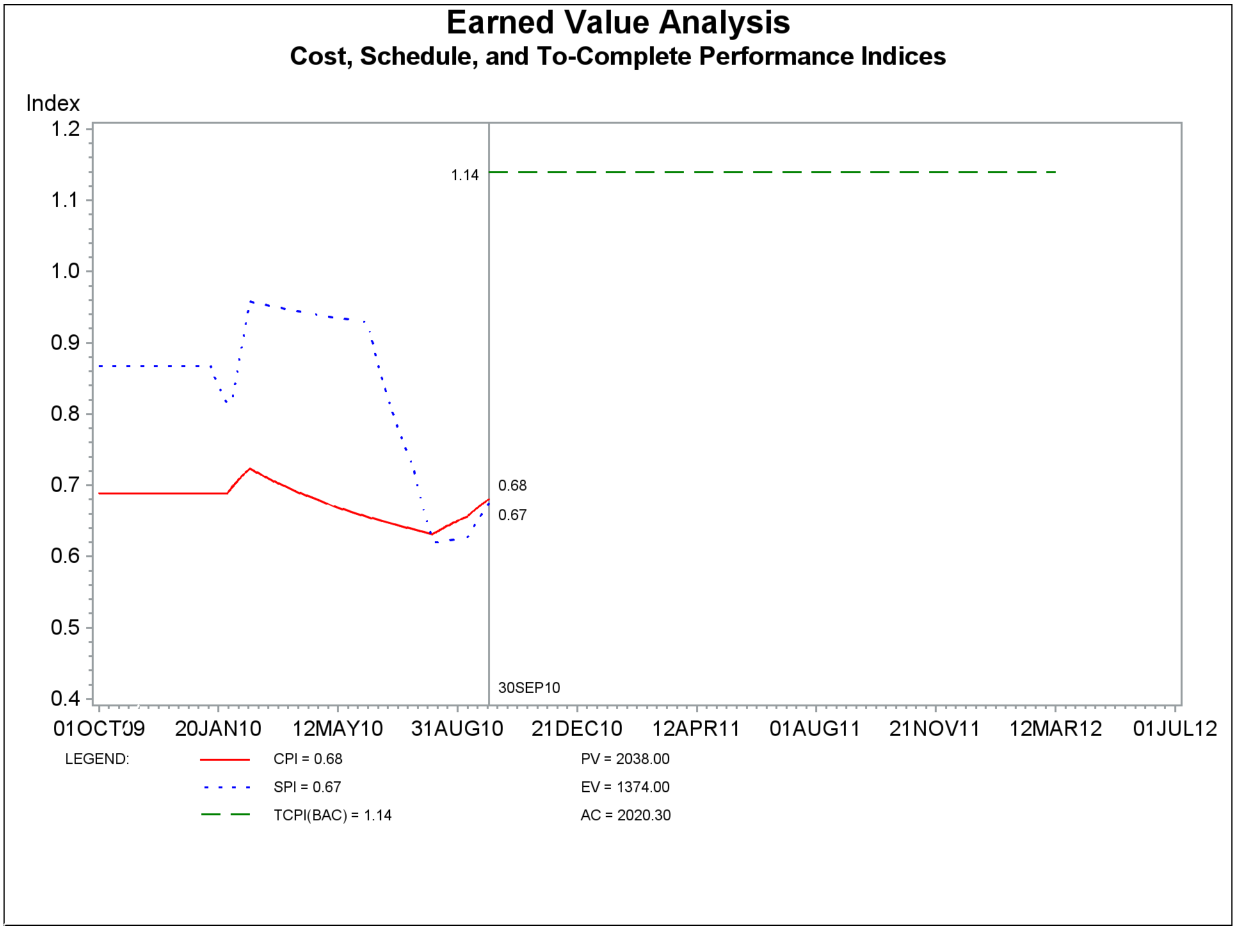
Output 10.1.10 %EVG_SCHEDULE_PLOT: Projected Completion Date



The %EVG_INDEX_PLOT macro is then used to produce the plot in Output 10.1.11.

```
%evg_index_plot;
```

Output 10.1.11 %EVG_INDEX_PLOT: Cost and Schedule Performance Index

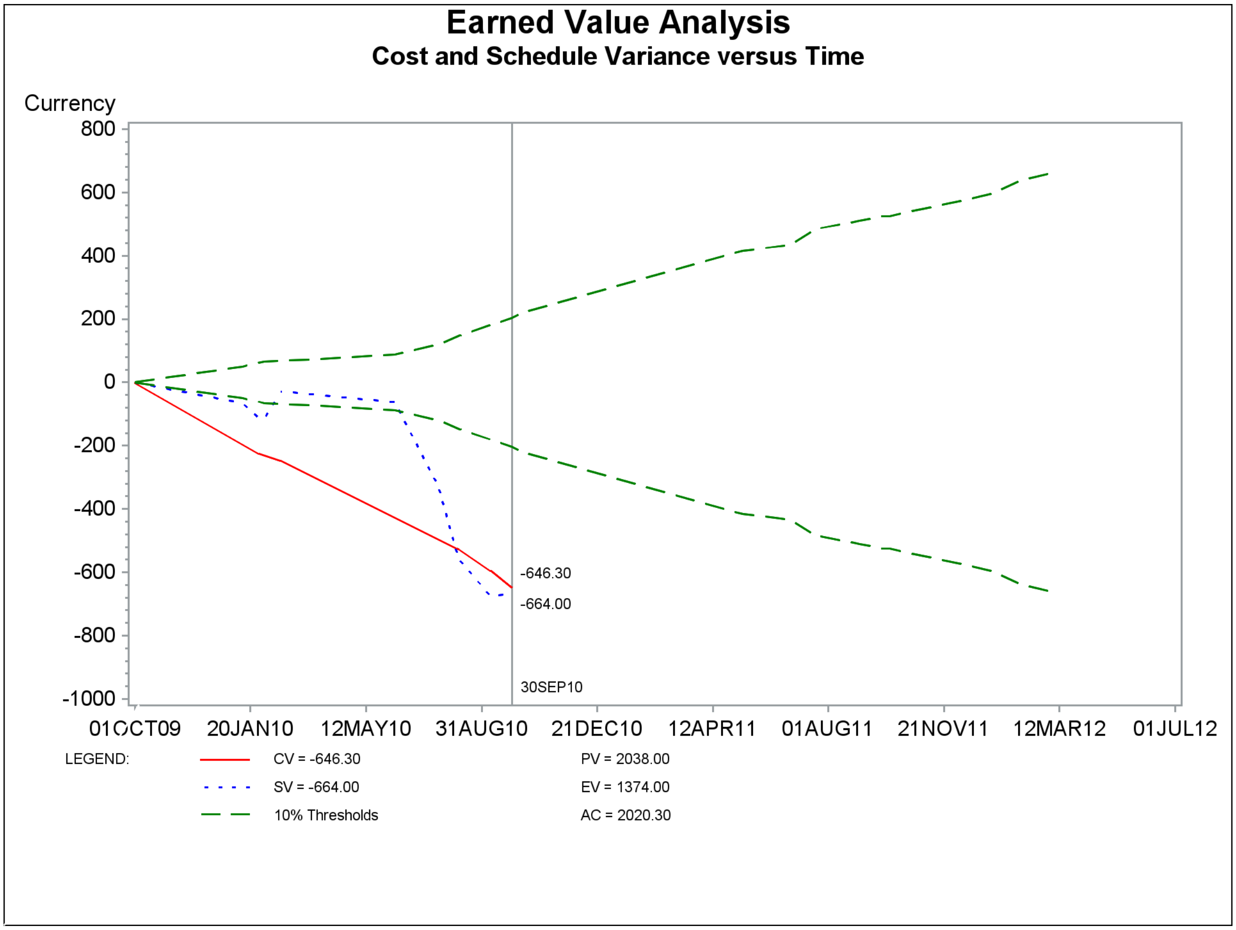


The plot in Output 10.1.11 shows that the performance factor must be increased from 0.68 to 1.14 in order to stay within the budget.

The %EVG_VARIANCE_PLOT macro is used to produce the plot in Output 10.1.12.

```
%evg_variance_plot;
```

Output 10.1.12 %EVG_VARIANCE_PLOT: Cost and Schedule Variance



The plot in Output 10.1.12 shows that both the cost and schedule variance have strayed outside of the area between the 10% threshold plots of the planned value.

Finally, the %EVG_GANTT_CHART macro is used to produce the Gantt chart shown in Output 10.1.13.

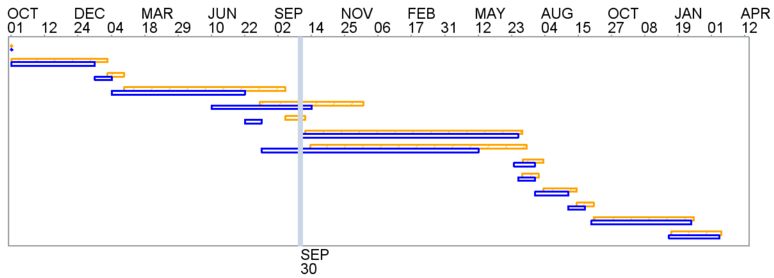
```
%evg_gantt_chart(
    plansched=iout1,
    revisesched=updsched,
    activity=activity,
    start=start,
    finish=finish,
    duration=duration,
    timenow='30sep10'd,
    id=pv ev ac cv cvp sv svp,
    height=3,
    scale=1.5
);
```

Output 10.1.13 %EVG_GANTT_CHART: Cost and Schedule Variance by Task

Earned Value Analysis

Gantt Chart Showing Variances and Performance Indices

Activity	BCWS	BCWP	ACWP	CV	CV%	SV	SV%
S	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PD	500.00	500.00	726.00	-226.00	-45.20	0.00	0.00
PDR	150.00	150.00	172.20	-22.20	-14.80	0.00	0.00
FD	300.00	300.00	629.30	-329.30	-109.77	0.00	0.00
PM	681.59	304.00	332.80	-28.80	-9.47	-377.59	-55.40
FDR	150.00	120.00	160.00	-40.00	-33.33	-30.00	-20.00
FP	0.00	0.00	0.00	0.00	0.00	0.00	0.00
FC	256.41	0.00	0.00	0.00	0.00	-256.41	-100.00
DA	0.00	0.00	0.00	0.00	0.00	0.00	0.00
FRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00
IA	0.00	0.00	0.00	0.00	0.00	0.00	0.00
RR	0.00	0.00	0.00	0.00	0.00	0.00	0.00
T	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TV	0.00	0.00	0.00	0.00	0.00	0.00	0.00



LEGEND: Revised Schedule, Baseline Schedule, Milestone

Example 10.2: Construction Project

This example illustrates a home building multiproject. The cost structure of the multiproject consists of both rates and fixed costs. This example also demonstrates how to accommodate multiple status dates. The initial schedule is shown in Output 10.2.1.

Output 10.2.1 Initial Schedule GBSCHED

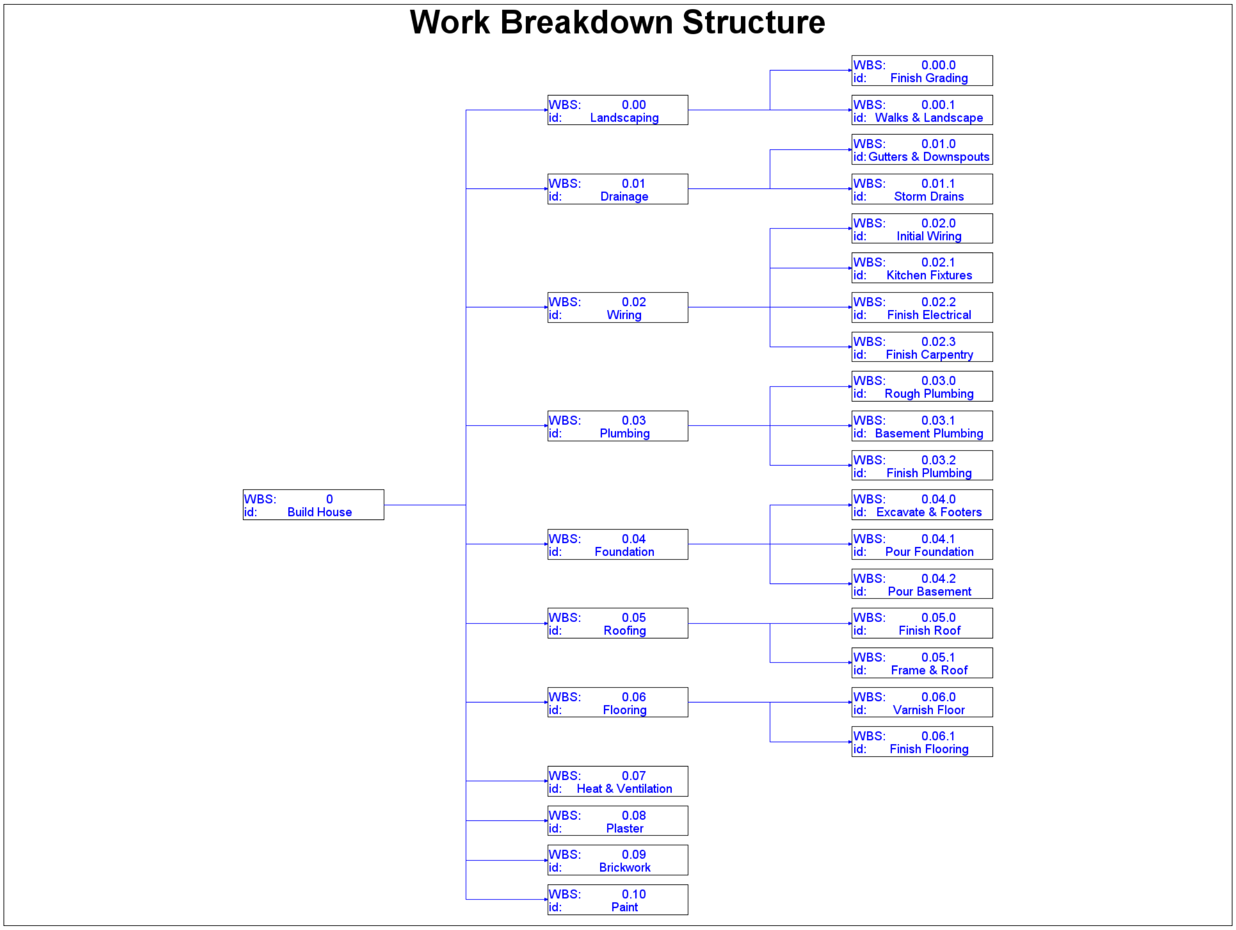
Planned Construction Schedule

Obs	Activity	WBS Code	Duration	Scheduled Start	Scheduled Finish
1	Build House	0	1	04SEP10	12OCT10
2	Landscaping	0.00	1	29SEP10	10OCT10
3	Finish Grading	0.00.0	2	29SEP10	30SEP10
4	Walks & Landscape	0.00.1	5	06OCT10	10OCT10
5	Drainage	0.01	1	28SEP10	29SEP10
6	Gutters & Downspouts	0.01.0	1	28SEP10	28SEP10
7	Storm Drains	0.01.1	1	29SEP10	29SEP10
8	Wiring	0.02	1	14SEP10	05OCT10
9	Initial Wiring	0.02.0	2	14SEP10	15SEP10
10	Kitchen Fixtures	0.02.1	1	03OCT10	03OCT10
11	Finish Electrical	0.02.2	1	04OCT10	04OCT10
12	Finish Carpentry	0.02.3	3	03OCT10	05OCT10
13	Plumbing	0.03	1	10SEP10	04OCT10
14	Rough Plumbing	0.03.0	3	11SEP10	13SEP10
15	Basement Plumbing	0.03.1	1	10SEP10	10SEP10
16	Finish Plumbing	0.03.2	2	03OCT10	04OCT10
17	Foundation	0.04	1	04SEP10	15SEP10
18	Excavate & Footers	0.04.0	4	04SEP10	07SEP10
19	Pour Foundation	0.04.1	2	08SEP10	09SEP10
20	Pour Basement	0.04.2	2	14SEP10	15SEP10
21	Roofing	0.05	1	10SEP10	27SEP10
22	Finish Roof	0.05.0	2	26SEP10	27SEP10
23	Frame & Roof	0.05.1	4	10SEP10	13SEP10
24	Flooring	0.06	1	30SEP10	12OCT10
25	Varnish Floor	0.06.0	2	11OCT10	12OCT10
26	Finish Flooring	0.06.1	3	30SEP10	02OCT10
27	Heat & Ventilation	0.07	4	16SEP10	19SEP10
28	Plaster	0.08	10	20SEP10	29SEP10
29	Brickwork	0.09	6	20SEP10	25SEP10
30	Paint	0.10	3	06OCT10	08OCT10

The Work Breakdown Structure for the project is given in Output 10.2.2; this chart was created using the %EVG_WBS_CHART macro.

```
%evg_wbs_chart(
   structure=gbsched,
   activity=activity,
   project=project,
   id=wbs_code id,
   rotate=N,
   rotatetext=N,
   defid=N
);
```

Output 10.2.2 %EVG_WBS_CHART: Work Breakdown Structure



Output 10.2.3 lists the budgeted costs for each task.

Output 10.2.3 Budgeted Cost Rates GBASE

Planned Construction Costs

Obs	Activity	cost	Start Pct./Weights	rate
1	Build House	0	0	0
2	Landscaping	0	0	0
3	Drainage	0	0	0
4	Wiring	0	0	0
5	Plumbing	0	0	0
6	Foundation	0	0	0
7	Roofing	0	0	0
8	Flooring	0	0	0
9	Heat & Ventilation	325	50	40
10	Plaster	14500	1,2,3,4	20
11	Brickwork	9500	1,2,3	45
12	Paint	3250	50	30
13	Finish Grading	425	25	25
14	Walks & Landscape	2475	25	25
15	Gutters & Downspouts	1200	50	15
16	Storm Drains	150	25	15
17	Initial Wiring	575	25	45
18	Kitchen Fixtures	375	25	25
19	Finish Electrical	550	25	50
20	Finish Carpentry	1450	25	20
21	Rough Plumbing	1025	25	30
22	Basement Plumbing	1000	25	50
23	Finish Plumbing	350	25	20
24	Excavate & Footers	5250	50	20
25	Pour Foundation	1500	25	15
26	Pour Basement	950	50	20
27	Finish Roof	725	50	40
28	Frame & Roof	9500	25	35
29	Varnish Floor	750	25	15
30	Finish Flooring	425	50	25

Note that the "Storm Drains" task, in the sixteenth observation, costs one quarter (25%) of \$150 upon initiation and continues at a rate of \$15/day. When complete, the balance (75%) of the \$150 is charged. Weighted milestones are specified for the "Plaster" activity, given in Observation 10. At the start, one tenth $\left(\frac{1}{1+2+3+4}\right)$ of \$14,500, or \$1,450, is incurred. The rate is \$20/day. When the task is one third complete, another fifth $\left(\frac{2}{1+2+3+4}\right)$ of \$14,500, or \$2,900, is incurred. At completion, the cost is two fifths $\left(\frac{4}{1+2+3+4}\right)$ of \$14,500, or \$5,800.

The %EVA_PLANNED_VALUE macro is next invoked to compute the periodic planned value.

```
%eva_planned_value(
   plansched=gbsched,
   activity=activity,
   start=start,
   finish=finish,
   budgetcost=gbase,
   rate=rate,
   cost=cost,
   spct=spct,
   taskpv=bout,
   pv=gbcost
);
```

The periodic planned value data set is shown in Output 10.2.4.

Output 10.2.4 %EVA_PLANNED_VALUE: Periodic Data Set

Daily Planned Value

Obs	Period Identifier	PV Rate
1	04SEP10	2645.00
2	05SEP10	20.00
3	06SEP10	20.00
4	07SEP10	2645.00
5	08SEP10	390.00
6	09SEP10	1140.00
7	10SEP10	3460.00
8	11SEP10	321.25
9	12SEP10	65.00
10	13SEP10	7958.75
11	14SEP10	683.75
12	15SEP10	971.25
13	16SEP10	202.50
14	17SEP10	40.00
15	18SEP10	40.00
16	19SEP10	202.50
17	20SEP10	3098.33
18	21SEP10	65.00
19	22SEP10	3231.67
20	23SEP10	2965.00
21	24SEP10	65.00
22	25SEP10	4815.00
23	26SEP10	4772.50
24	27SEP10	422.50
25	28SEP10	1235.00
26	29SEP10	6116.25
27	30SEP10	581.25
28	01OCT10	25.00
29	02OCT10	237.50
30	03OCT10	890.00
31	04OCT10	902.50
32	05OCT10	1107.50
33	06OCT10	2298.75
34	07OCT10	55.00
35	08OCT10	1680.00
36	09OCT10	25.00
37	10OCT10	1881.25
38	11OCT10	202.50
39	12OCT10	577.50

Notice that the TASKPV= parameter has been used to ultimately pass the planned activity duration and costs to %EVA_EARNED_VALUE. Assume that the schedule has been updated to reflect actual start and finish times, as of the status date September 15, 2010. The updated schedule is shown in Output 10.2.5.

Output 10.2.5 Updated Schedule GASCHED

Updated Construction Schedule

Obs	Activity	WBS Code	Duration	Scheduled Start	Scheduled Finish
1	Build House	0	1	04SEP10	17OCT10
2	Landscaping	0.00	1	04OCT10	15OCT10
3	Drainage	0.01	1	03OCT10	04OCT10
4	Wiring	0.02	1	19SEP10	10OCT10
5	Plumbing	0.03	1	15SEP10	09OCT10
6	Foundation	0.04	1	04SEP10	20SEP10
7	Roofing	0.05	1	15SEP10	02OCT10
8	Flooring	0.06	1	05OCT10	17OCT10
9	Heat & Ventilation	0.07	4	21SEP10	24SEP10
10	Plaster	0.08	10	25SEP10	04OCT10
11	Brickwork	0.09	6	25SEP10	30SEP10
12	Paint	0.10	3	11OCT10	13OCT10
13	Finish Grading	0.00.0	2	04OCT10	05OCT10
14	Walks & Landscape	0.00.1	5	11OCT10	15OCT10
15	Gutters & Downspouts	0.01.0	1	03OCT10	03OCT10
16	Storm Drains	0.01.1	1	04OCT10	04OCT10
17	Initial Wiring	0.02.0	2	19SEP10	20SEP10
18	Kitchen Fixtures	0.02.1	1	08OCT10	08OCT10
19	Finish Electrical	0.02.2	1	09OCT10	09OCT10
20	Finish Carpentry	0.02.3	3	08OCT10	10OCT10
21	Rough Plumbing	0.03.0	3	16SEP10	18SEP10
22	Basement Plumbing	0.03.1	1	15SEP10	15SEP10
23	Finish Plumbing	0.03.2	2	08OCT10	09OCT10
24	Excavate & Footers	0.04.0	4	04SEP10	11SEP10
25	Pour Foundation	0.04.1	2	12SEP10	14SEP10
26	Pour Basement	0.04.2	2	19SEP10	20SEP10
27	Finish Roof	0.05.0	2	01OCT10	02OCT10
28	Frame & Roof	0.05.1	4	15SEP10	18SEP10
29	Varnish Floor	0.06.0	2	16OCT10	17OCT10
30	Finish Flooring	0.06.1	3	05OCT10	07OCT10

The updated cost rates are given in Output 10.2.6.

Output 10.2.6 Updated Cost Rates GACT

Updated Construction Costs

Obs	Activity	Rate	Cost	Start Pct./Weights
1	Build House	0	0	0
2	Landscaping	0	0	0
3	Drainage	0	0	0
4	Wiring	0	0	0
5	Plumbing	0	0	0
6	Foundation	0	0	0
7	Roofing	0	0	0
8	Flooring	0	0	0
9	Heat & Ventilation	40	325	50
10	Plaster	20	14500	1,2,3,4
11	Brickwork	45	9500	1,2,3
12	Paint	30	3250	50
13	Finish Grading	25	425	25
14	Walks & Landscape	25	2475	25
15	Gutters & Downspouts	15	1200	50
16	Storm Drains	15	150	25
17	Initial Wiring	45	575	25
18	Kitchen Fixtures	25	375	25
19	Finish Electrical	50	550	25
20	Finish Carpentry	20	1450	25
21	Rough Plumbing	30	1025	25
22	Basement Plumbing	50	1000	25
23	Finish Plumbing	20	350	25
24	Excavate & Footers	30	6350	50
25	Pour Foundation	30	2700	25
26	Pour Basement	20	950	50
27	Finish Roof	40	725	50
28	Frame & Roof	35	9500	25
29	Varnish Floor	15	750	25
30	Finish Flooring	25	425	50

The %EVA_EARNED_VALUE macro can then be called as follows:

```
%eva_earned_value(
   revisesched=gasched,
   activity=activity,
   start=start,
   finish=finish,
   actualcost=gact,
   rate=rate,
   cost=cost,
   spct=spct,
   taskpv=bout,
   ev=gacost
);
```

The periodic earned value data set that is generated by %EVA_EARNED_VALUE is shown in Output 10.2.7.

Output 10.2.7 %EVA_EARNED_VALUE: Periodic Data Set

Daily Earned Value and Revised Cost

Obs	Period Identifier	EV Rate	AC Rate
1	04SEP10	2635.00	3205.00
2	05SEP10	10.00	30.00
3	06SEP10	10.00	30.00
4	07SEP10	10.00	30.00
5	08SEP10	10.00	30.00
6	09SEP10	10.00	30.00
7	10SEP10	10.00	30.00
8	11SEP10	2635.00	3205.00
9	12SEP10	385.00	705.00
10	13SEP10	10.00	30.00
11	14SEP10	1135.00	2055.00
12	15SEP10	3460.00	3460.00
13	16SEP10	321.25	321.25
14	17SEP10	65.00	65.00
15	18SEP10	7958.75	7958.75
16	19SEP10	683.75	683.75
17	20SEP10	971.25	971.25
18	21SEP10	202.50	202.50
19	22SEP10	40.00	40.00
20	23SEP10	40.00	40.00
21	24SEP10	202.50	202.50
22	25SEP10	3098.33	3098.33
23	26SEP10	65.00	65.00
24	27SEP10	3231.67	3231.67
25	28SEP10	2965.00	2965.00
26	29SEP10	65.00	65.00
27	30SEP10	4815.00	4815.00
28	01OCT10	4772.50	4772.50
29	02OCT10	422.50	422.50
30	03OCT10	1235.00	1235.00
31	04OCT10	6116.25	6116.25
32	05OCT10	581.25	581.25
33	06OCT10	25.00	25.00
34	07OCT10	237.50	237.50
35	08OCT10	890.00	890.00
36	09OCT10	902.50	902.50
37	10OCT10	1107.50	1107.50
38	11OCT10	2298.75	2298.75
39	12OCT10	55.00	55.00
40	13OCT10	1680.00	1680.00
41	14OCT10	25.00	25.00
42	15OCT10	1881.25	1881.25
43	16OCT10	202.50	202.50
44	17OCT10	577.50	577.50

The BUDGETCOST= parameter has been employed to capture the planned duration and costs for each activity. Next, the %EVA_METRICS macro is used to produce statistics for the entire project. For illustrative purposes, a range of times from the start of the project to the revised projected end date is used. Typically, only actual status dates would be used; in this case, perhaps September 4, 2010 and September 15, 2010. Unless otherwise noted, the latter is the assumption for the remainder of the macros in this example.

```
%eva_metrics(
   pv=gbcost,
   ev=gacost,
   timenow='04SEP10'd '15SEP10'd '01OCT10'd '17OCT10'd
);
```

The output listing is given in Output 10.2.8.

Output 10.2.8 %EVA_METRICS: Summary Statistics

Earned Value Analysis

Metric	September 4, 2010	September 15, 2010	October 1, 2010	October 17, 2010
Percent Complete	4.54	17.78	68.59	100.00
PV (Planned Value)	2645.00	20320.00	48197.50	58055.00
EV (Earned Value)	2635.00	10320.00	39817.50	58055.00
AC (Actual Cost)	3205.00	12840.00	42337.50	60575.00
CV (Cost Variance)	-570.00	-2520.00	-2520.00	-2520.00
CV%	-21.63	-24.42	-6.33	-4.34
SV (Schedule Variance)	-10.00	-10000.00	-8380.00	0.00
SV%	-0.38	-49.21	-17.39	0.00
CPI (Cost Performance Index)	0.82	0.80	0.94	0.96
SPI (Schedule Performance Index)	1.00	0.51	0.83	1.00
BAC (Budget At Completion)	58055.00	58055.00	58055.00	58055.00
EAC (Revised Estimate At Completion)	60575.00	60575.00	60575.00	60575.00
EAC (Overrun to Date)	58625.00	60575.00	60575.00	60575.00
EAC (Cumulative CPI)	70613.39	72231.22	61729.23	60575.00
EAC (Cumulative CPI X SPI)	70869.21	129780.85	65810.42	60575.00
ETC (Estimate To Complete)*	67408.39	59391.22	19391.73	0.00
VAC (Variance At Completion)*	-12558.39	-14176.22	-3674.23	-2520.00
VAC%*	-21.63	-24.42	-6.33	-4.34
TCPI (BAC) (To-Complete Performance Index)	1.01	1.06	1.16	0.00
TCPI (EAC) (To-Complete Performance Index)*	0.82	0.80	0.94	0.94

*** The CPI form of the EAC is used.**

Observe that the Estimate To Complete (ETC) and Schedule Performance Index (SPI) converge to 0 and 1, respectively, over time. Also, the Actual Cost (AC) agrees with the various Estimates At Completion (EAC's) at the projected completion date.

Next, the %EVA_TASK_METRICS macro is used to show an activity-level view of the progress of the project.

```
%eva_task_metrics(
   activity=id,
   plansched=gbsched,
   revisesched=gasched,
   start=start,
   finish=finish,
   budgetcost=gbase,
   actualcost=gact,
   cost=cost,
   spct=spct,
   rate=rate,
   timenow='15SEP10'd,
   aggregate=Y
);
```

The AGGREGATE= parameter is specified in order to roll up the values with respect to the project hierarchy. The output from this macro is shown in Output 10.2.9 and Output 10.2.10.

Output 10.2.9 %EVA_TASK_METRICS: Metrics by Activity

Earned Value Analysis by Activity
as of September 15, 2010

Obs	WBS Code	PV	EV	AC
1	0	20320.00	10320.00	12840.00
2	0.00	0.00	0.00	0.00
3	0.01	0.00	0.00	0.00
4	0.02	665.00	0.00	0.00
5	0.03	2165.00	1050.00	1050.00
6	0.04	7850.00	6860.00	9380.00
7	0.05	9640.00	2410.00	2410.00
8	0.06	0.00	0.00	0.00
9	0.07	0.00	0.00	0.00
10	0.08	0.00	0.00	0.00
11	0.09	0.00	0.00	0.00
12	0.10	0.00	0.00	0.00
13	0.00.0	0.00	0.00	0.00
14	0.00.1	0.00	0.00	0.00
15	0.01.0	0.00	0.00	0.00
16	0.01.1	0.00	0.00	0.00
17	0.02.0	665.00	0.00	0.00
18	0.02.1	0.00	0.00	0.00
19	0.02.2	0.00	0.00	0.00
20	0.02.3	0.00	0.00	0.00
21	0.03.0	1115.00	0.00	0.00
22	0.03.1	1050.00	1050.00	1050.00
23	0.03.2	0.00	0.00	0.00
24	0.04.0	5330.00	5330.00	6590.00
25	0.04.1	1530.00	1530.00	2790.00
26	0.04.2	990.00	0.00	0.00
27	0.05.0	0.00	0.00	0.00
28	0.05.1	9640.00	2410.00	2410.00
29	0.06.0	0.00	0.00	0.00
30	0.06.1	0.00	0.00	0.00

Output 10.2.10 %EVA_TASK_METRICS: Metrics by Activity (continued)

Earned Value Analysis by Activity
as of September 15, 2010

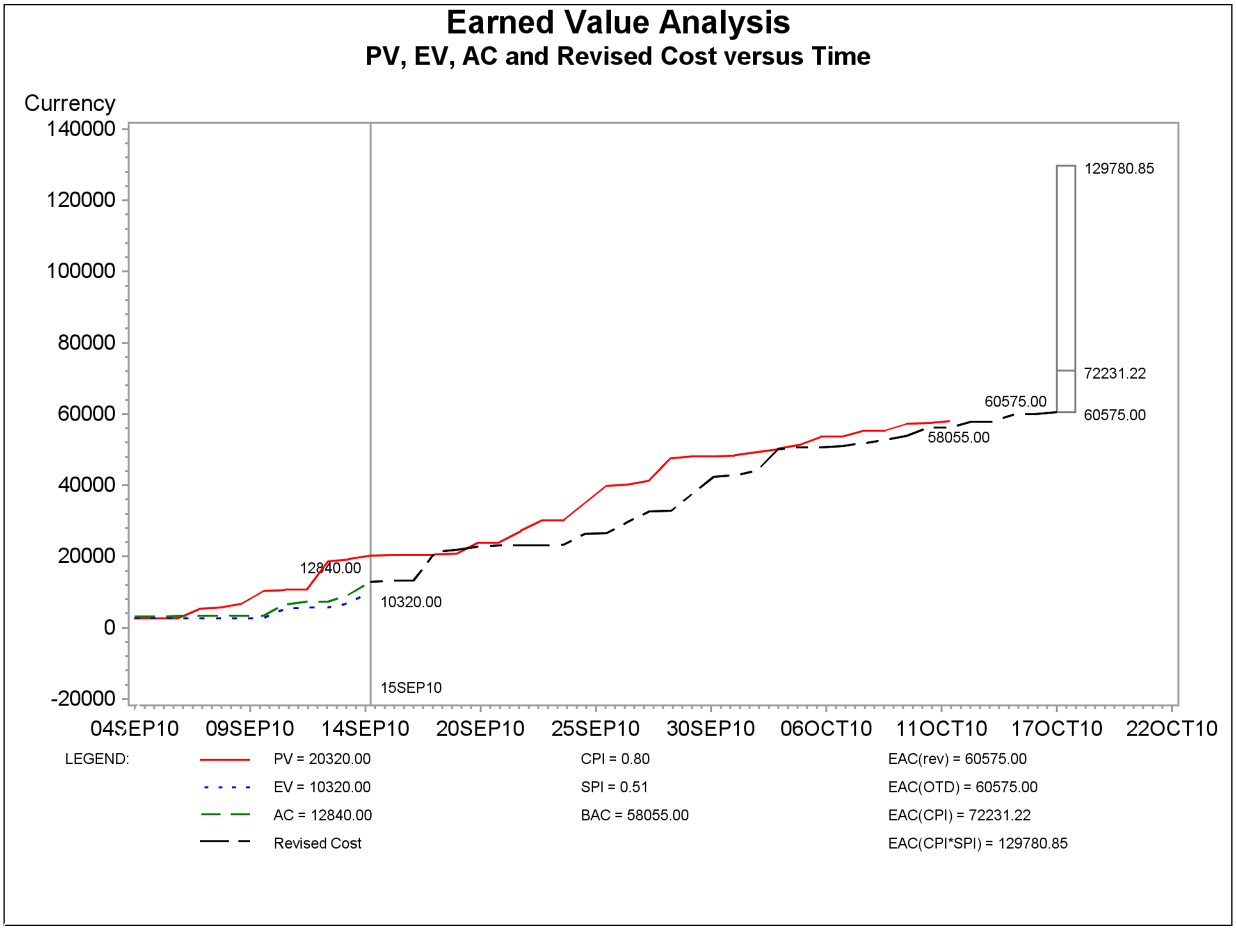
Obs	CV	CV%	SV	SV%	CPI	SPI
1	-2520.00	-24.42	-10000.00	-49.21	0.80	0.51
2	0.00	0.00	0.00	0.00	.	.
3	0.00	0.00	0.00	0.00	.	.
4	0.00	0.00	-665.00	-100.00	.	0.00
5	0.00	0.00	-1115.00	-51.50	1.00	0.48
6	-2520.00	-36.73	-990.00	-12.61	0.73	0.87
7	0.00	0.00	-7230.00	-75.00	1.00	0.25
8	0.00	0.00	0.00	0.00	.	.
9	0.00	0.00	0.00	0.00	.	.
10	0.00	0.00	0.00	0.00	.	.
11	0.00	0.00	0.00	0.00	.	.
12	0.00	0.00	0.00	0.00	.	.
13	0.00	0.00	0.00	0.00	.	.
14	0.00	0.00	0.00	0.00	.	.
15	0.00	0.00	0.00	0.00	.	.
16	0.00	0.00	0.00	0.00	.	.
17	0.00	0.00	-665.00	-100.00	.	0.00
18	0.00	0.00	0.00	0.00	.	.
19	0.00	0.00	0.00	0.00	.	.
20	0.00	0.00	0.00	0.00	.	.
21	0.00	0.00	-1115.00	-100.00	.	0.00
22	0.00	0.00	0.00	0.00	1.00	1.00
23	0.00	0.00	0.00	0.00	.	.
24	-1260.00	-23.64	0.00	0.00	0.81	1.00
25	-1260.00	-82.35	0.00	0.00	0.55	1.00
26	0.00	0.00	-990.00	-100.00	.	0.00
27	0.00	0.00	0.00	0.00	.	.
28	0.00	0.00	-7230.00	-75.00	1.00	0.25
29	0.00	0.00	0.00	0.00	.	.
30	0.00	0.00	0.00	0.00	.	.

Next, the %EVG_COST_PLOT macro is called to show the Planned Value (PV), Earned Value (EV), Actual Cost (AC), and revised cost plots.

```
%evg_cost_plot;
```

The plot is shown in Output 10.2.11.

Output 10.2.11 %EVG_COST_PLOT, PV, EV, AC and Revised Cost

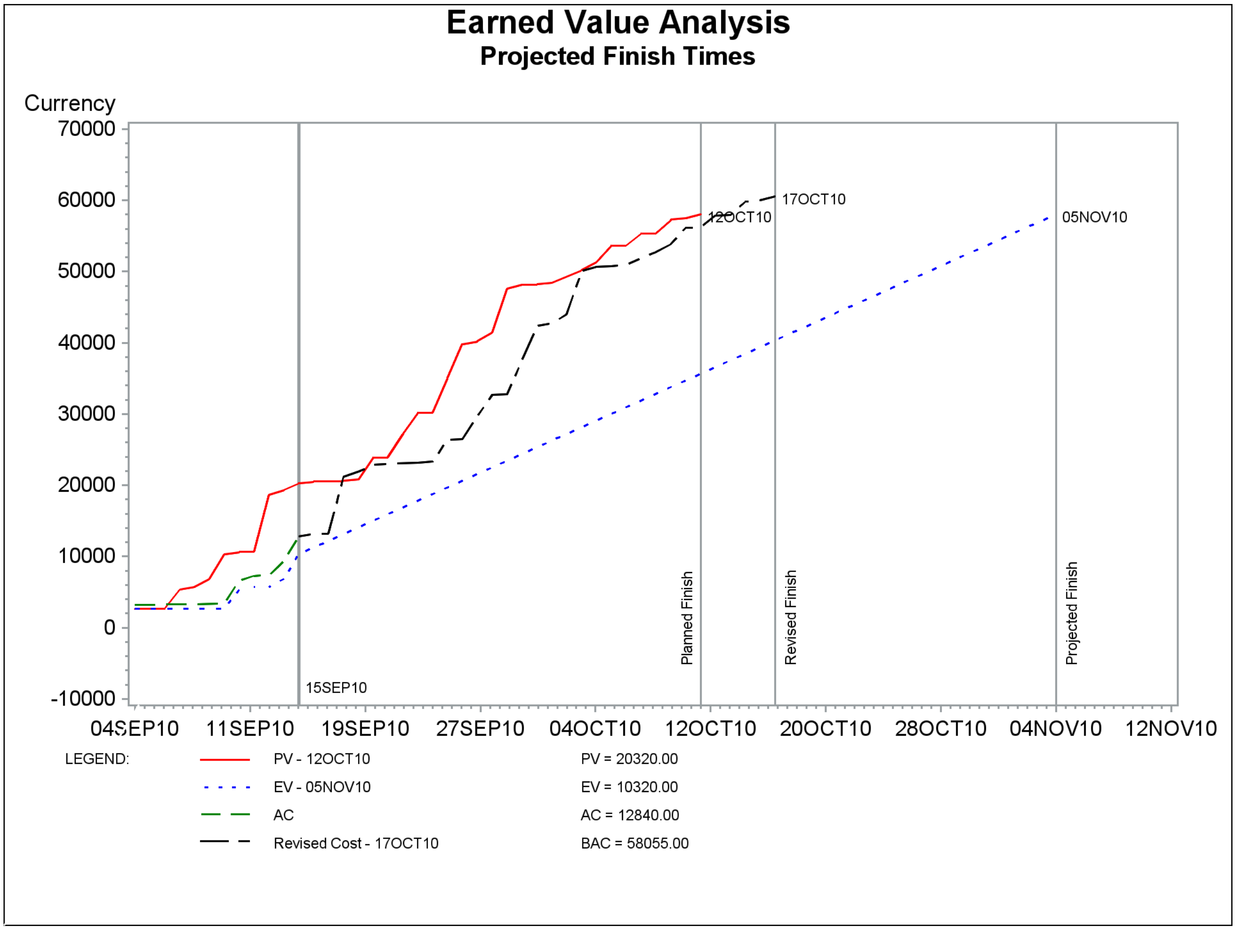


The %EVG_SCHEDULE_PLOT macro is used to show different estimates of the project completion date.

```
%evg_schedule_plot;
```

The plot is shown in Output 10.2.12.

Output 10.2.12 %EVG_SCHEDULE_PLOT: Estimated Completion Dates

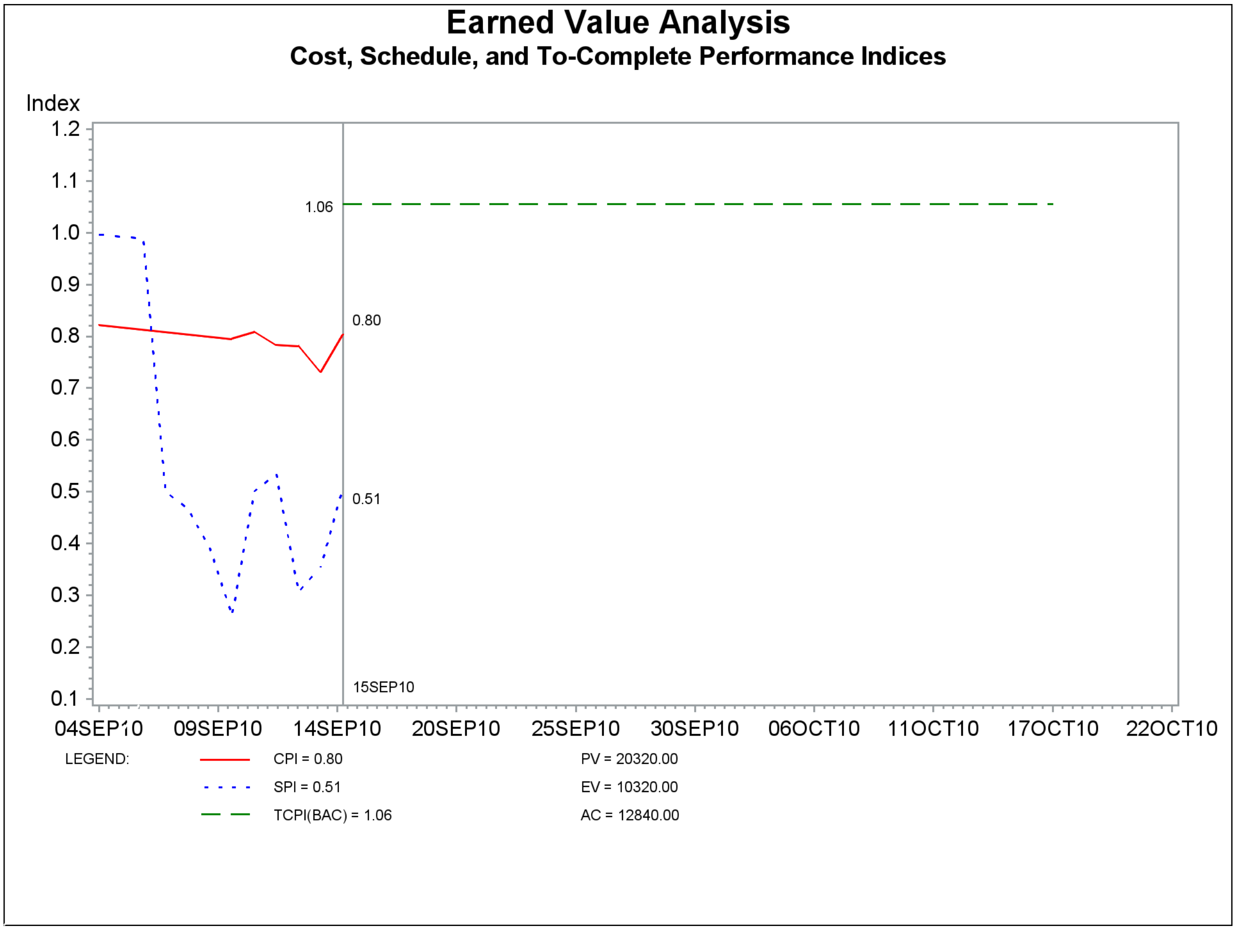


The %EVG_INDEX_PLOT macro is used to graphically display the performance indices for the project.

```
%evg_index_plot;
```

The plot is shown in Output 10.2.13. As in the previous example, the cost performance needs to increase from 0.8 to 1.06 in order to stay within the budget.

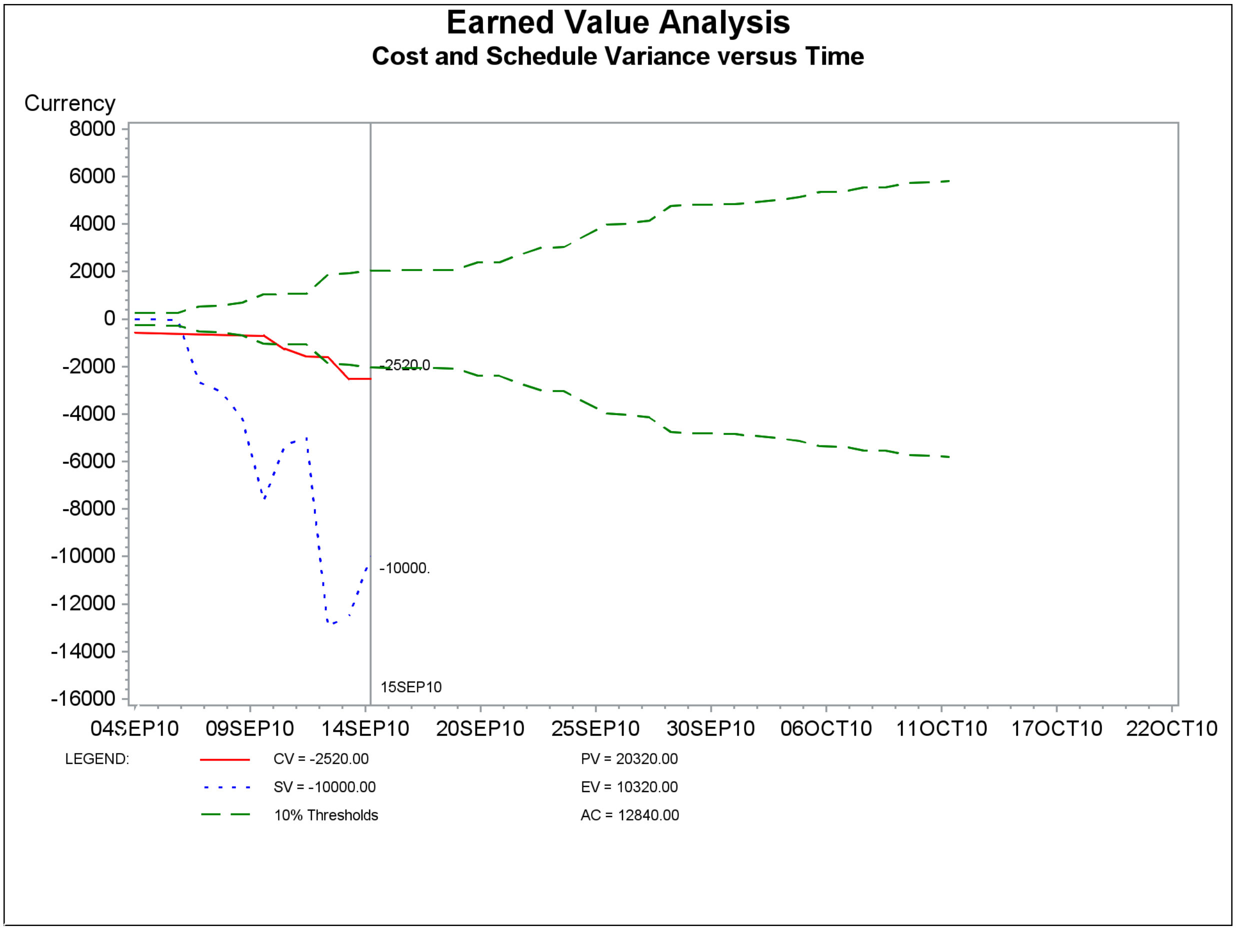
Output 10.2.13 %EVG_INDEX_PLOT: CPI, SPI, and TCPI



The Cost and Schedule Variance for the project is shown in Output 10.2.14 using the following call to the %EVG_VARIANCE_PLOT macro:

```
%evg_variance_plot;
```

Output 10.2.14 %EVG_VARIANCE_PLOT: Cost and Schedule Variance



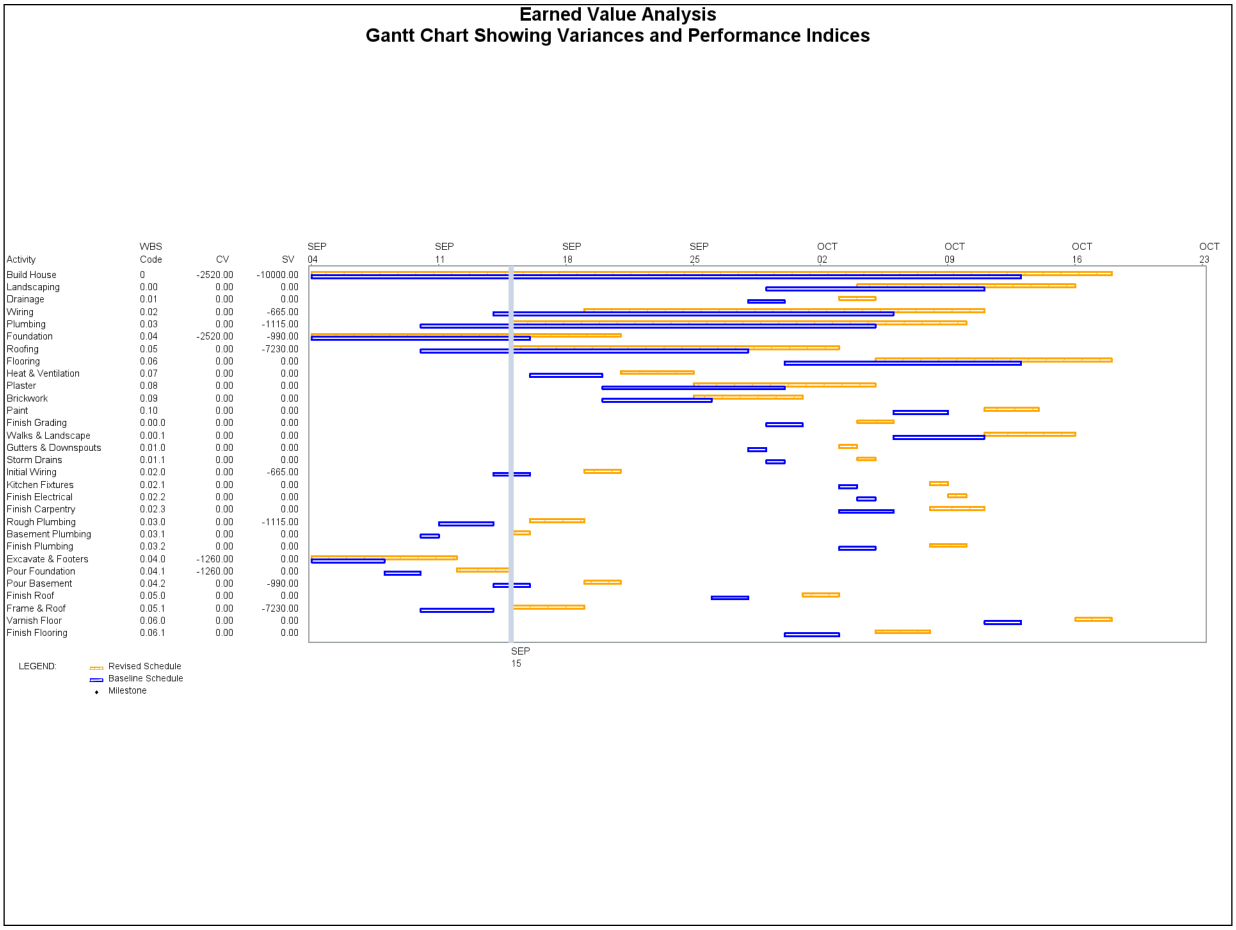
While the Cost Variance is just outside the 10% threshold, the Schedule Variance is significantly outside and should be a warning signal that the project is slipping significantly. The plan called for $20,320 of work to be completed, but only $10,320 has been accomplished so far (roughly 50% of the goal).

Finally, the %EVG_GANTT_CHART macro is used to show a Gantt view of the project along with some metrics by task. In this case, only the Work Breakdown Structure code and Cost Variance (CV) and Schedule Variance (SV) are selected.

```
%evg_gantt_chart(
   activity=id,
   plansched=gbsched,
   revisesched=gasched,
   start=start,
   finish=finish,
   timenow='15SEP10'd,
   id=wbs cv sv,
   height=3,
   scale=40
);
```

The resulting Gantt chart is shown in Output 10.2.15.

Output 10.2.15 %EVG_GANTT_CHART: Cost and Schedule Variance by Task



References

Fleming, Q. W., and Koppelman, J. M. (2000). *Earned Value Project Management*. 2nd ed. Newtown Square, PA: Project Management Institute.

Project Management Institute (1996). *A Guide to the Project Management Body of Knowledge*. Upper Darby, PA: Project Management Institute.

SAS Institute Inc. (1993). *SAS/OR Software: Project Management Examples*. Cary, NC: SAS Institute Inc.

Appendix A
Glossary of Project Management Terms

Glossary

A

activity

an element of work performed during the course of a project. An activity normally has an expected duration, an expected cost, and expected resource requirements. Activities are often subdivided into tasks.

activity delay

the maximum amount of time that an activity can be delayed due to lack of resources.

activity-on-arrow (AOA)

see *arrow diagramming method*.

activity-on-node (AON)

see *precedence diagramming method*.

activity priority

a priority value that is assigned to activities to provide an ordering for activities that are waiting for resources (during resource-constrained scheduling).

activity splitting

The act of dividing activities into segments during resource allocation. In some cases, preemption of activity segments can free a resource to be used by a more critical activity.

actual cost of work performed (ACWP)

total costs incurred (direct and indirect) in accomplishing work during a given time period. See also *earned value*.

actual finish date (AF)

the calendar date when work on an activity actually ended. The AF date must be prior to the timenow date.

actual start date (AS)

the calendar date when work on an activity actually began. The AS date must be prior to the timenow date.

aggregation

the use of activity resource requirements to calculate total resource needs rather than to constrain the project schedule. Normally, resource requirements are used to perform resource-constrained scheduling.

alignment type

an identification of the type of constraint that is associated with a target date. The following types are available:

- finish on
- finish on or after
- finish on or before
- start on
- start on or after
- start on or before
- mandatory start
- mandatory finish

arrow

the graphic representation of an activity. See also *arrow diagramming method*.

arrow diagramming method

a network diagramming technique in which activities are represented by arrows. The tail of the arrow represents the start of the activity, and the head represents the finish of the activity (the length of the arrow does not represent the expected duration of the activity). Activities are connected at points called nodes (usually drawn as small circles) to illustrate the sequence in which the activities are expected to be performed. See also *precedence diagramming method*.

as-of date

see *timenow date*.

B

backward pass

the calculation of late finish dates and late start dates for the uncompleted portions of all network activities, determined by working backwards through the network logic from the project end date. The end date can be specified, although it is usually calculated in a forward pass.

baseline schedule

a project schedule that consists of baseline start and finish dates, which represent an estimated or expected schedule, or both. This schedule is often derived from an initial set of early, late, or scheduled finish dates. Typically, once a baseline schedule is established, it does not change over the course of a project.

baseline finish date (BF)

the calendar date when work on an activity is scheduled to end. This date is usually estimated, or it can be derived from the early, late, or scheduled finish dates. Typically, once a baseline schedule is established, it does not change over the course of the project.

baseline start date (BS)

the calendar date when work on an activity is scheduled to begin. This date is usually estimated, or it can be derived from the early, late, or scheduled start dates. Typically, once a baseline schedule is established, it does not change over the course of the project.

budget at completion (BAC)

the estimated total cost of the project when done.

budgeted cost of work performed (BCWP)

the sum of the approved cost estimates (including any overhead allocation) for activities (or portions of activities) that are completed during a given period (usually project-to-date). See also *earned value*.

budgeted cost of work scheduled (BCWS)

the sum of the approved cost estimates (including any overhead allocation) for activities (or portions of activities) that are scheduled to be performed during a given period (usually project-to-date). See also *earned value*.

C

calendar

a method of identifying project work days that can be altered so that weekends, holidays, vacation, weather days, and so forth are not included.

cost performance index (CPI)

the ratio of budgeted costs to actual costs (BCWP/ACWP). The CPI is often used to predict the magnitude of a possible cost overrun using the following formula: original cost estimate/CPI = projected cost at completion. See also *earned value*.

cost variance (CV)

(1) any difference between the estimated cost of an activity and the actual cost of an activity.

(2) in earned value, BCWP less ACWP.

critical activity

any activity on the critical path.

critical path

the series of activities of a project that determines the earliest completion of the project. The critical path generally changes from time to time as activities are completed ahead of or behind schedule. The critical path is usually defined as those activities with total float less than or equal to zero. See also *critical path method*.

critical path method (CPM)

a network analysis technique used to predict project duration by analyzing which sequence of activities (which path) has the least amount of scheduling flexibility (the least amount of total float). Early dates are calculated by means of a forward pass using a specified start date. Late dates are calculated by means of a backward pass starting from a specified completion date (usually the calculated project early finish date of the forward pass).

cycle

see *loop*.

D

data date

see *timenow date*.

dependency

see *logical relationship*.

duration

the number of work periods (not including holidays or other nonworking periods) required to complete an activity or set of activities. All activity durations are specified with the same duration unit.

duration unit

the unit of time that each activity in the project lasts. The following choices are available:

- second
- minute
- hour
- day
- weekday
- week
- month
- qtr
- year

E

early finish date (EF)

in the critical path method, the earliest possible point in time at which the uncompleted portions of an activity (or the project) can finish, based on the network logic and any schedule constraints. Early finish dates can change as the project progresses and changes are made to the project plan.

early start date (ES)

in the critical path method, the earliest possible point in time at which the uncompleted portions of an activity (or the project) can start, based on the network logic and any schedule constraints. Early start dates can change as the project progresses and changes are made to the project plan.

earned value (EV)

(1) a method for measuring project performance that compares the amount of work that was planned with what was actually accomplished to determine whether cost and schedule performance are as planned. See also *actual cost of work performed*, *budgeted cost of work performed*, *budgeted cost of work scheduled*, *cost variance*, *cost performance index*, *schedule variance*, and *schedule performance index*.

(2) the budgeted cost of work performed, for an activity or group of activities.

earned value analysis

see definition (1) under *earned value*.

effort

the number of labor units required to complete an activity or other project element. Usually expressed as staffhours, staffdays, or staffweeks. Should not be confused with duration.

estimate at completion (EAC)

the expected total cost of an activity, group of activities, or the project when the defined scope of work has been completed. Most techniques for forecasting EAC include some adjustment of the original cost estimate based on project performance to date. Also called "estimated at completion." Often shown as EAC = Actuals-to-date + ETC. See also *earned value* and *estimate to complete*.

estimate to complete (ETC)

the expected additional cost needed to complete an activity, a group of activities, or the project. Most techniques for forecasting ETC include some adjustment to the original cost estimate based on project performance to date. Also called "estimated to complete." See also *earned value* and *estimate at completion*.

F

float

see *total float*.

forward pass

the calculation of the early start and early finish dates for the uncompleted portions of all network activities. See also *backward pass*.

free float (FF)

the amount of time an activity can be delayed without delaying the early start of any immediate successor activities. See also *total float*.

G

Gantt chart

a graphic representation of work activities shown by a time-scaled bar chart.

graphical evaluation and review technique (GERT)

a network analysis technique that allows for conditional and probabilistic treatment of logical relationships (that is, some activities might not be performed).

H

holiday

a period of time within the project timeframe when work cannot be scheduled. Holidays can be assigned to one or more calendars.

L

lag

a modification of a logical relationship that directs a delay of the successor task. For example, in a finish-to-start dependency with a 10-day lag, the successor activity can start 10 days after the predecessor has finished. See also *lead*.

late finish date (LF)

in the critical path method, the latest possible point in time that an activity can be completed without delaying a specified milestone (usually the project finish date).

late start date (LS)

in the critical path method, the latest possible point in time that an activity can begin without delaying a specified milestone (usually the project finish date).

lead

a modification of a logical relationship that allows an acceleration of the successor task. For example, in a finish-to-start dependency with a 10-day lead, the successor activity can start 10 days before the predecessor has finished. See also *lag*.

logic

the collection of activity dependencies that make up a project network diagram.

logic diagram

see *network diagram*.

logical relationship

a dependency between two project activities. The four possible types of logical relationships are:

- finish-to-start—the “from” activity must finish before the “to” activity can start.
- finish-to-finish—the “from” activity must finish before the “to” activity can finish.
- start-to-start—the “from” activity must start before the “to” activity can start.
- start-to-finish—the “from” activity must start before the “to” activity can finish.

Finish-to-start is defined as the standard (or default) logical relationship.

loop

a network path that passes the same node twice. Loops cannot be analyzed by using traditional network analysis techniques such as CPM and PERT. Loops are allowed in GERT.

M

maximum number of segments

the maximum number of segments that an activity can be split into when activity splitting is allowed.

milestone

a significant event in the project, usually completion of a major deliverable.

minimum segment duration

the minimum duration of a segment of an activity when activity splitting is allowed.

N

near-critical activity

an activity that has low total float.

network

see *network diagram*.

network analysis

the process of identifying early and late start and finish dates for the uncompleted portions of project activities. See also *critical path method*, *program evaluation and review technique*, and *graphical evaluation and review technique*.

network diagram

a schematic display of the logical relationships of project activities. Always drawn from left to right to reflect project chronology. Often incorrectly referred to as a “PERT chart.”

network logic

see *logic*.

network path

any continuous series of connected activities that make up a project network diagram.

node

one of the defining points of a network; a junction point joined to some or all of the other dependency lines. Also, the graphic representation of an activity. See also *arrow diagramming method* and *precedence diagramming method*.

nonstandard logical relationship

a dependency between two project activities that is not the standard finish-to-start relationship. See *logical relationship* for the four possible types of relationships.

O

organizational breakdown structure (OBS)

a depiction of the project organization arranged so as to relate work packages to organizational units.

overlap

see *lead*.

P

parent task

see *supertask*.

path

a set of sequentially connected activities in a project network diagram.

path float

see *total float*.

percent complete

an estimate, expressed as a percent, of the amount of work that has been completed on an activity or group of activities.

PERT chart

a specific type of project network diagram. See *program evaluation and review technique*.

precedence diagramming method (PDM)

a network diagramming technique in which activities are represented by boxes (or nodes). Activities are linked together by precedence relationships to show the sequence in which the activities are to be performed.

precedence relationship

the term used in the precedence diagramming method for a logical relationship. In current usage, precedence relationship, logical relationship, and dependency are widely used interchangeably regardless of the diagramming method in use.

predecessor activity

any activity that exists on a common path with the activity in question and occurs before the activity in question.

preemption

see *activity splitting*.

program evaluation and review technique (PERT)

an event-oriented network analysis technique that is used to estimate project duration when there is a high degree of uncertainty with the individual activity duration estimates. PERT applies the critical path method to a weighted average duration estimate.

project

a temporary endeavor undertaken to create a unique product or service. A project consists of one or more activities.

project management

the application of knowledge, skills, tools, and techniques to project activities in order to meet or exceed stakeholder needs and expectations from a project.

project management body of knowledge (PMBOK)

an inclusive term that describes the sum of knowledge within the profession of project management. As with other professions such as law, medicine, and accounting, the body of knowledge rests with the practitioners and academics who apply and advance it. The PMBOK includes proven, traditional practices that are widely applied in addition to innovative and advanced ones that have seen more limited use.

project network diagram

see *network diagram*.

project schedule

the planned dates for performing activities and the planned dates for meeting milestones.

R

remaining duration

the amount of time needed to complete an activity.

resource-constrained scheduling

the scheduling of activities in a project with the knowledge of certain resource constraints and requirements. This process adjusts activity scheduled start and finish dates to conform to resource availability and use.

resource leveling

any form of network analysis in which scheduling decisions (start and finish dates) are driven by resource management concerns (for example, limited resource availability or difficult-to-manage changes in resource levels).

S

schedule

see *project schedule*.

schedule analysis

see *network analysis*.

schedule performance index (SPI)

the ratio of work that is performed to work that is scheduled (BCWP/BCWS). See *earned value*.

schedule variance

(1) any difference between the scheduled completion of an activity and the actual completion of that activity.

(2) in earned value, BCWP less BCWS.

scheduled finish date (SF)

the date when the activity is scheduled to be completed using the resource-constrained scheduling process.

scheduled start date (SS)

the date when the activity is scheduled to begin using the resource-constrained scheduling process. This date is equal to or greater than the early start date.

slack

term used in PERT for float (see also *total float*).

subtask

an activity that is contained within a supertask.

successor activity

any activity that exists on a common path with the activity in question and occurs after the activity in question.

supertask

an aggregate or summary activity that contains one or more activities (subtasks) such that no subtask can begin until the supertask has begun. The supertask cannot end until all of the subtasks have ended.

T

target date

date that is used to constrain the start or finish of an activity. The type of constraint is identified by an alignment type.

task

see *activity*.

timenow date

the calendar date that separates actual (historical) data from future (scheduled) data.

total float (TF)

the amount of time that an activity can be delayed from its early start without delaying the project finish date. Total float is a mathematical calculation and can change as the project progresses and changes are made to the project plan. Also called "float," "slack," and "path float." See also *free float*.

W

work breakdown structure (WBS)

a deliverable-oriented grouping of project elements that organizes and defines the total scope of the project. Each descending level represents an increasingly detailed definition of a project component. Project components can be products or services.

work packages

a deliverable at the lowest level of the work breakdown structure. A work package can be divided into activities.

workshift

one or more pairs of on/off working times that define the valid working periods within a single day.

References

Project Management Institute (1996). *A Guide to the Project Management Body of Knowledge*. Upper Darby, PA: Project Management Institute.

Appendix B
Glossary of Earned Value Management Terms

Glossary

A

AC

see *actual cost*.

ACWP

see *actual cost*.

actual cost

total costs incurred that must relate to whatever cost was budgeted within the *planned value* and *earned value* (which can sometimes be direct labor hours alone, direct costs alone, or all costs including indirect costs) in accomplishing work during a given time period.

actual cost of work performed

see *actual cost*.

B

BAC

see *budget at completion*.

BCWP

see *earned value*.

BCWS

see *planned value*.

budget at completion

the sum of the total budgets for a project.

budgeted cost of work performed

see *earned value*.

budgeted cost of work scheduled

see *planned value*.

C

CPI

see *cost performance index*.

CV

see *cost variance*.

CV%

see *cost variance percentage*.

cost performance index (CPI)

the cost efficiency ratio of *earned value* to *actual cost*. CPI is often used to predict the magnitude of a possible cost overrun by using the following formula: $\frac{BAC}{CPI}$ = projected cost at completion. $CPI = \frac{EV}{AC}$.

cost variance (CV)

the difference between the *earned value* of an activity and the *actual cost* of that activity; that is, $CV = EV - AC$.

cost variance percentage (CV%)

cost variance relative to earned value: $CV\% = \frac{CV}{EV}$.

E

EAC

see *estimate at completion*.

ETC

see *estimate to complete*.

EV

see *earned value*.

earned value

the physical work that is accomplished plus the authorized budget for this work. The sum of the approved cost estimates (which might include overhead allocation) for activities (or portions of activities) that are completed during a given period (usually project-to-date). This term was previously called the budgeted cost of work performed (BCWP) for an activity or group of activities.

earned value management

a method for integrating scope, schedule, and resources, and for measuring project performance. It compares the value of work that was planned, the value of work that was actually completed, and the amount of money that was actually spent, to determine if cost and schedule performance are as planned.

estimate at completion (EAC)

the expected total cost of an activity, a group of activities, or the project when the defined scope of work has been completed. Most techniques for producing the EAC include some adjustment of the original cost estimate, based on actual project performance to date. This metric is derived from the ETC as follows: $EAC = AC + ETC$. See *estimate to complete*.

estimate to complete (ETC)

the expected additional cost needed to complete an activity, a group of activities, or the project. Most techniques for producing the ETC factor in the project performance to date. The revised ETC, or ETC_{rev}, is the cumulative cost from the status date to the project completion date, according to the revised schedule and costs. The other three formulations used in this document are as follows:

$$ETC_{OTD} = BAC - EV$$

$$ETC_{CPI} = \frac{BAC - EV}{CPI}$$

$$ETC_{CPI \times SPI} = \frac{BAC - EV}{CPI \times SPI}$$

P

PV

see *planned value*.

percentage complete

the percentage of total earned value that has been accumulated for a project:

$$percentage\ complete = \frac{EV}{BAC}$$

planned value

the physical work that is scheduled, plus the authorized budget to accomplish the scheduled work. Previously, this term was called the budgeted cost of work scheduled (BCWS).

S

SPI

see *schedule performance index*.

SV

see *schedule variance*.

SV%

see *schedule variance percentage*.

schedule performance index (SPI)

the schedule efficiency ratio of *earned value* that was accomplished against the *planned value*. The SPI describes what portion of the planned schedule was actually accomplished. $SPI = \frac{EV}{PV}$.

schedule variance (SV)

the difference between the *earned value* and the *planned value* of the project at any point in time; that is, $SV = EV - PV$.

schedule variance percentage (SV%)

schedule variance relative to planned value:

$$SV\% = \frac{SV}{PV}$$

T

TCPI

see *to-complete performance index*.

to-complete performance index (TCPI)

the cost of remaining work divided by the remaining budget or estimated budget:

$$TCPI(BAC) = \frac{BAC - EV}{BAC - AC}$$
$$TCPI(EAC) = \frac{BAC - EV}{EAC - AC}$$

V

VAC

see *variance at completion*.

VAC%

see *variance at completion percentage*.

variance at completion (VAC)

the difference between the original budget and the current estimate at completion: $VAC = BAC - EAC$.

variance at completion percentage (VAC%)

the variance at completion relative to the budget at completion: $VAC\% = \frac{VAC}{BAC}$.

W

WBS

see *work breakdown structure*.

work breakdown structure (WBS)

the project hierarchy.

Subject Index

Syntax Index