THE POWER TO KNOW.

SAS/OR® 15.1 User's Guide
Project Management
The CPM Procedure

This document is an individual chapter from *SAS/OR® 15.1 User's Guide: Project Management*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2018. *SAS/OR® 15.1 User's Guide: Project Management*. Cary, NC: SAS Institute Inc.

SAS/OR® 15.1 User's Guide: Project Management

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

November 2018

Chapter 4
The CPM Procedure

Contents

Overview: CPM Procedure **57**
Getting Started: CPM Procedure **58**
Syntax: CPM Procedure **63**
- Functional Summary 63
- PROC CPM Statement 67
- ACTIVITY Statement 72
- ACTUAL Statement 72
- ALIGNDATE Statement 75
- ALIGNTYPE Statement 75
- BASELINE Statement 76
- CALID Statement 77
- DURATION Statement 78
- HEADNODE Statement 78
- HOLIDAY Statement 79
- ID Statement 80
- PROJECT Statement 80
- RESOURCE Statement 82
- SUCCESSOR Statement 92
- TAILNODE Statement 93

Details: CPM Procedure **94**
- Scheduling Subject to Precedence Constraints 95
- Using the INTERVAL= Option 96
- Nonstandard Precedence Relationships 97
- Time-Constrained Scheduling 98
- Finish Milestones 100
- OUT= Schedule Data Set 101
- Multiple Calendars 103
- Baseline and Target Schedules 111
- Progress Updating 111
- Resource-Driven Durations and Resource Calendars 114
- Resource Usage and Allocation 115
- RESOURCEOUT= Usage Data Set 129
- RESOURCESCHED= Resource Schedule Data Set 133
- Multiproject Scheduling 133
- Macro Variable _ORCPM_ 136
- Input Data Sets and Related Variables 137

Missing Values in Input Data Sets . . . 139
FORMAT Specification . . . 141
Computer Resource Requirements . . . 141
Examples: CPM Procedure . . . **142**
Example 4.1: Activity-on-Node Representation . . . 144
Example 4.2: Activity-on-Arc Representation . . . 148
Example 4.3: Meeting Project Deadlines . . . 151
Example 4.4: Displaying the Schedule on a Calendar . . . 153
Example 4.5: Precedence Gantt Chart . . . 156
Example 4.6: Changing Duration Units . . . 157
Example 4.7: Controlling the Project Calendar . . . 161
Example 4.8: Scheduling around Holidays . . . 163
Example 4.9: CALEDATA and WORKDATA Data Sets . . . 169
Example 4.10: Multiple Calendars . . . 174
Example 4.11: Nonstandard Relationships . . . 183
Example 4.12: Activity Time Constraints . . . 188
Example 4.13: Progress Update and Target Schedules . . . 190
Example 4.14: Summarizing Resource Utilization . . . 195
Example 4.15: Resource Allocation . . . 201
Example 4.16: Using Supplementary Resources . . . 210
Example 4.17: INFEASDIAGNOSTIC Option and Aggregate Resource Type . . . 214
Example 4.18: Variable Activity Delay . . . 221
Example 4.19: Activity Splitting . . . 228
Example 4.20: Alternate Resources . . . 233
Example 4.21: PERT Assumptions and Calculations . . . 241
Example 4.22: Scheduling Course - Teacher Combinations . . . 244
Example 4.23: Multiproject Scheduling . . . 248
Example 4.24: Resource-Driven Durations and Resource Calendars . . . 258
Example 4.25: Resource-Driven Durations and Alternate Resources . . . 270
Example 4.26: Multiple Alternate Resources . . . 276
Example 4.27: Auxiliary Resources and Alternate Resources . . . 278
Example 4.28: Use of the SETFINISHMILESTONE Option . . . 281
Example 4.29: Negative Resource Requirements . . . 289
Example 4.30: Auxiliary Resources and Negative Requirements . . . 292
Example 4.31: Resource-Driven Durations and Negative Requirements . . . 296
Statement and Option Cross-Reference Tables . . . 301
References . . . **303**

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:

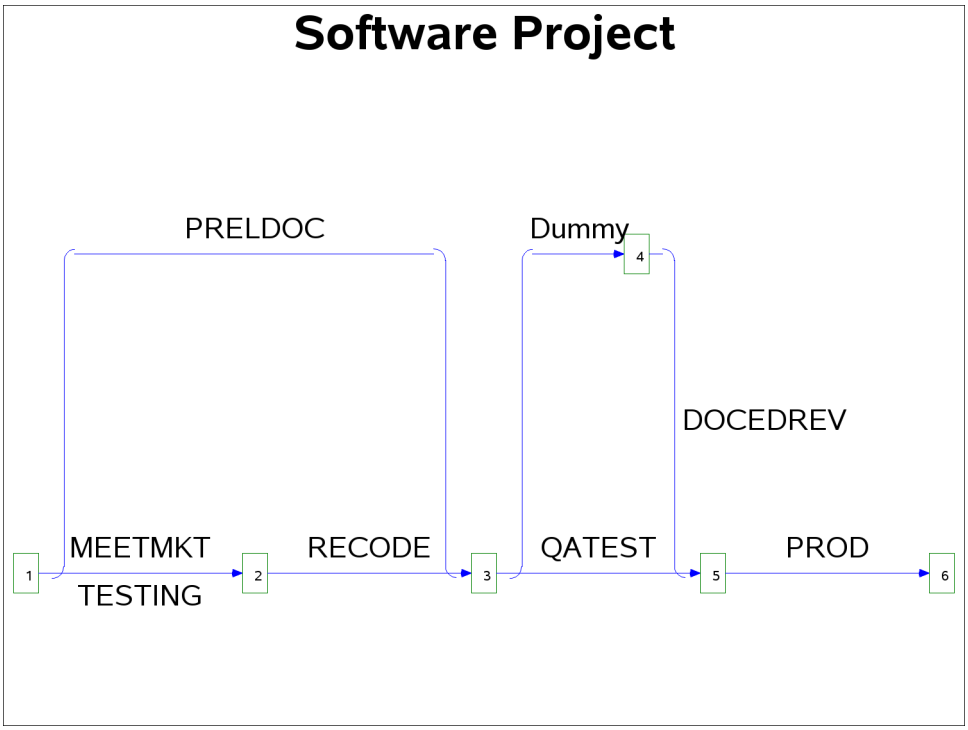
- The Schedule data set contains the early, late, baseline, resource-constrained, and actual schedules and any other activity-related information that is calculated by PROC CPM.
- The Resource Schedule data set contains the schedules for each resource used by an activity.
- The Usage data set contains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

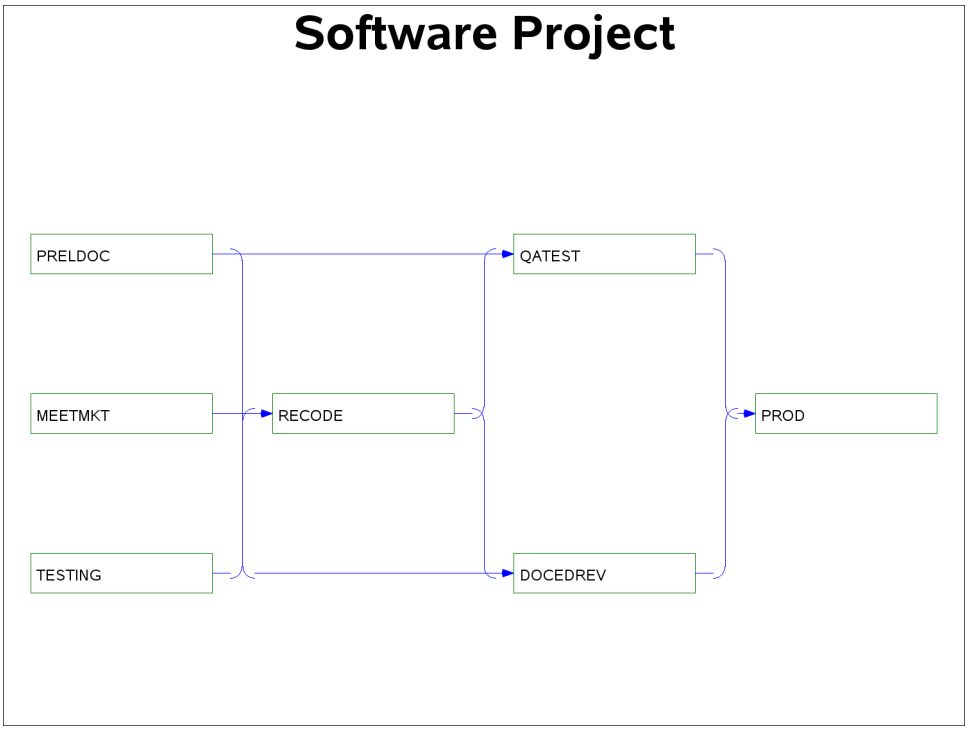
- Activity-On-Arc (AOA) or Activity-On-Edge (AOE) diagrams show the activities on the arcs or edges of the network. Figure 4.1 shows the AOA representation for the software project. This method of representing a project is known also as the arrow diagramming method (ADM). For projects represented in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

$$\text{T\_FLOAT} = \text{L\_START} - \text{E\_START} = \text{L\_FINISH} - \text{E\_FINISH}$$

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 103. 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 95 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 103 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 96 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*×2 if the network is specified in AOA format and to *nobs*×(*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*×*nresvar* ×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 101 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 116.

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 129.

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 100.

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 103 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 101. See the section "Progress Updating" on page 111 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 111 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 111 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 111 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 124 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 98 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 98 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 103 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 133 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

$$\text{PROJ\_DUR} = \text{E\_FINISH} - \text{E\_START}$$

then the SEPCRIT option sets

$$\text{L\_FINISH} \leq \text{E\_START} + \text{PROJ\_DUR}$$

while the USEPROJDUR option sets

$$\text{L\_FINISH} \leq \text{E\_START} + \text{DUR}$$

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 121 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 129 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 121 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 121. 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 127 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 125 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 129 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 122 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 114.

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 98, the statements needed to place time constraints on the activities are introduced. the section "Finish Milestones" on page 100 describes some options controlling the treatment of milestones.

The section "OUT= Schedule Data Set" on page 101 describes the format of the schedule output data set (the Schedule data set). The section "Multiple Calendars" on page 103 deals with calendar specifications for the different activities.

The section "Baseline and Target Schedules" on page 111 describes how you can save specific schedules as baseline or target schedules. The section "Progress Updating" on page 111 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 114 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 115 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 129 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 133, contains the individual resource schedules for each activity.

The section "Multiproject Scheduling" on page 133 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 136.

Table 4.9 in the section "Input Data Sets and Related Variables" on page 137 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 139 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 141 underlines the importance of associating the correct FORMAT specification with all the date-type variables, and the section "Computer Resource Requirements" on page 141 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 92 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 95, 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 95 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*B
FF	Finish-to-finish	*pef* + *ld* − *dur*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 114.

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 111 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 124.

If the PROJECT statement is used, some additional variables are added to the output data set. See the section "Schedule Data Set" on page 136 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 104. 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

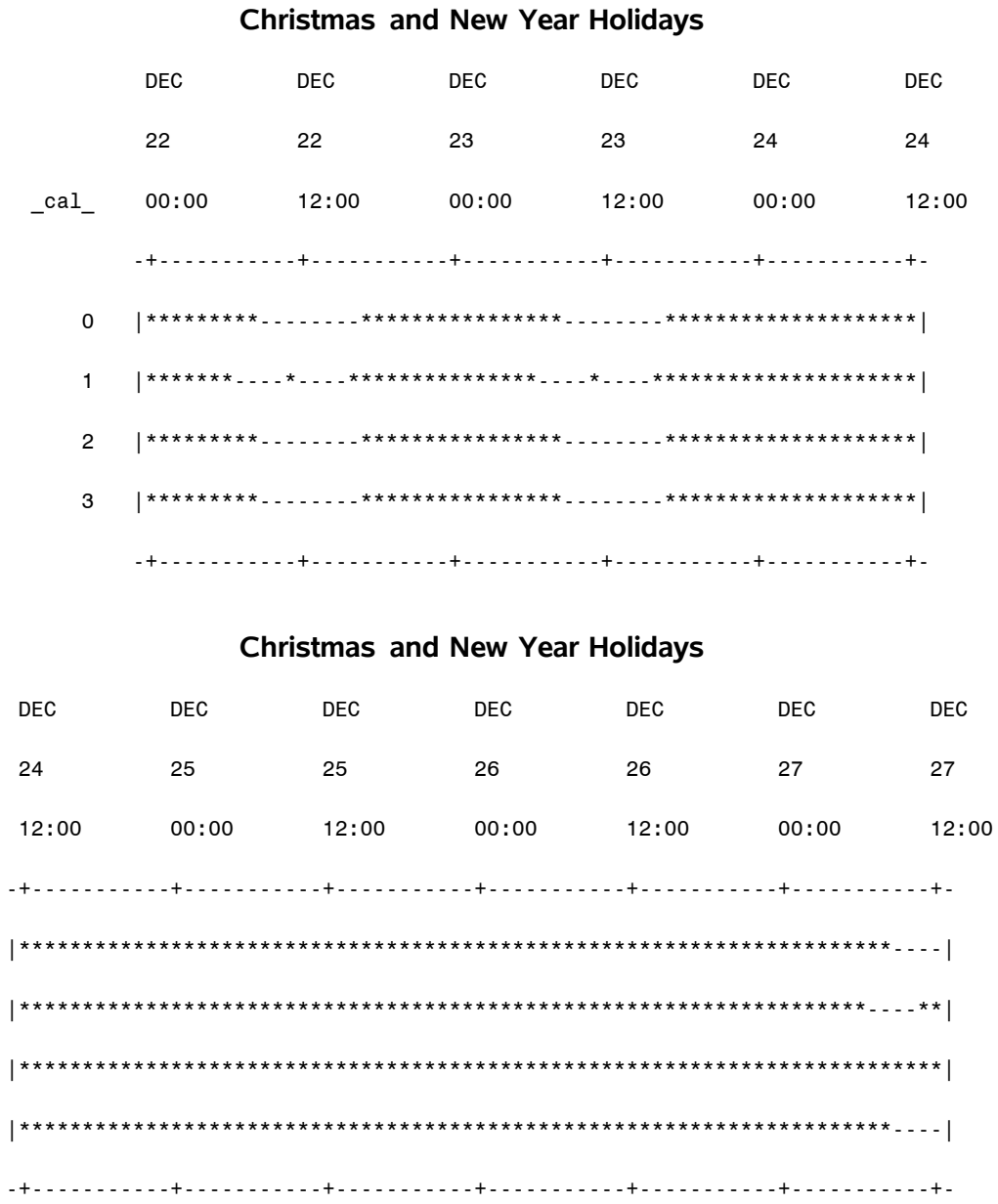


Figure 4.4 *continued*

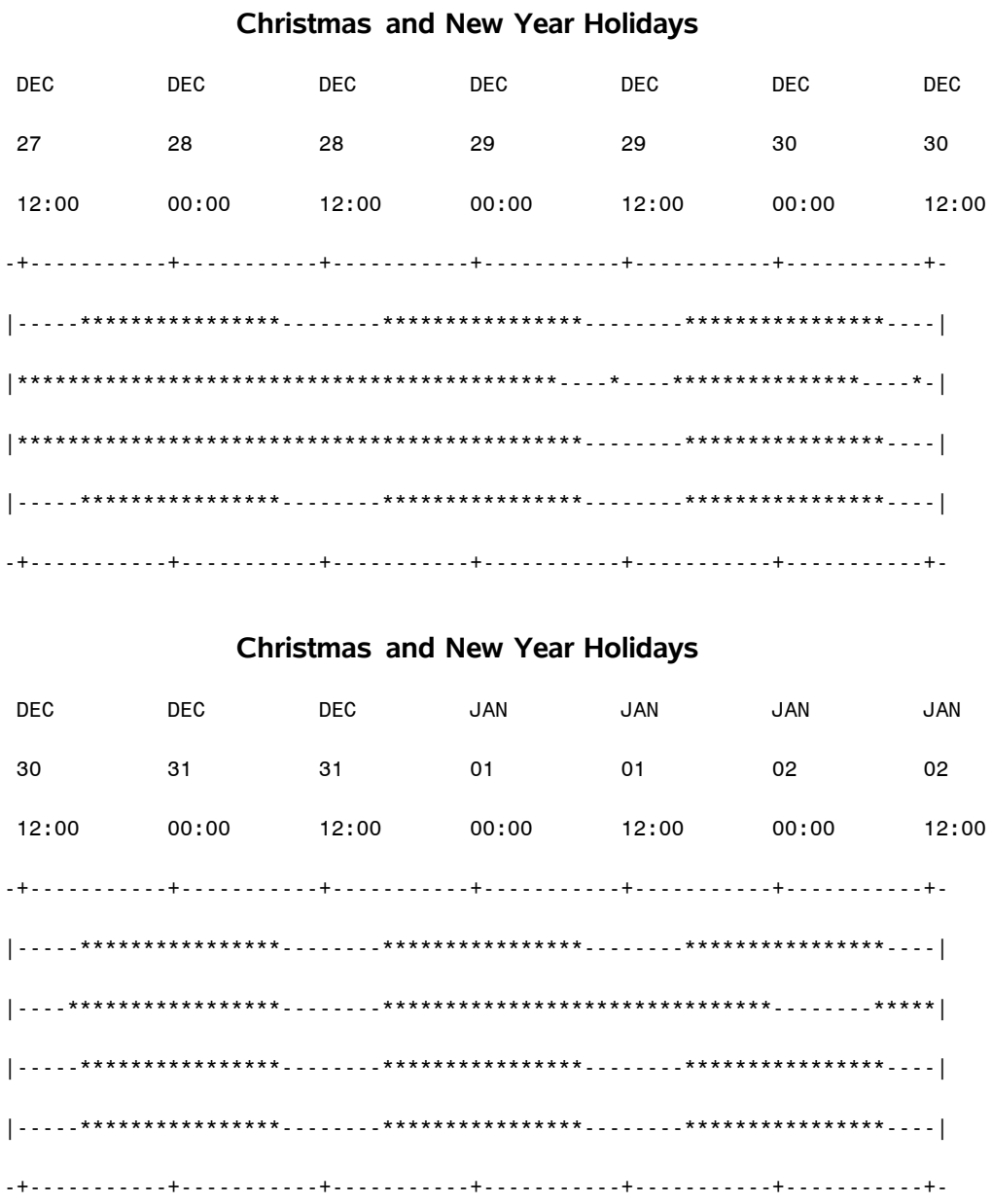


Figure 4.5 Vacation Time for Calendars 2 and 3

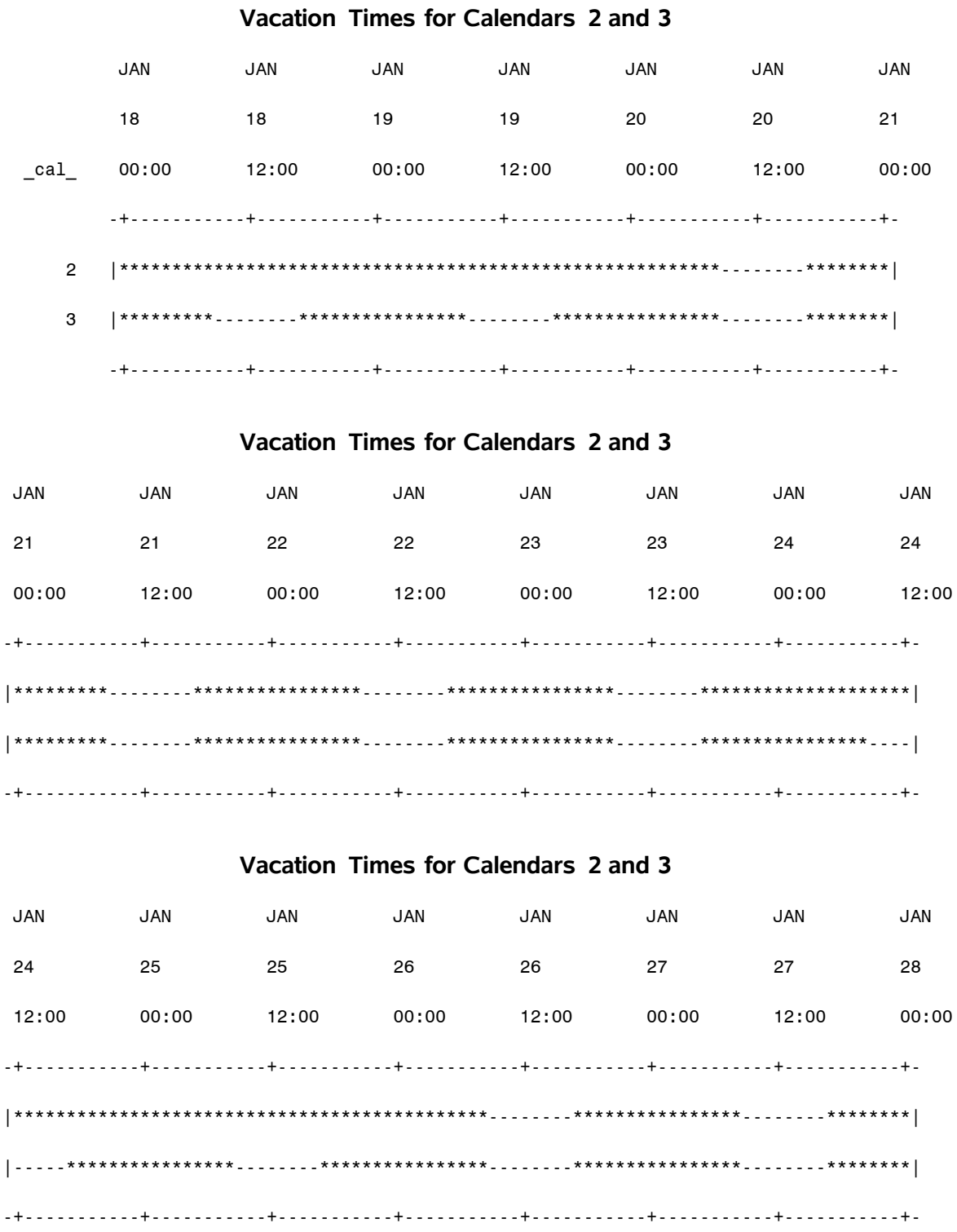


Figure 4.5 *continued*

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 76 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 116) 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 120, and the description of the format of the Resource data set is given in the section "RESOURCEIN= Input Data Set" on page 116, which follows. the section "Scheduling Method" on page 121 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 the primary 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 120). 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 122). 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 123 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 125 and the section "Auxiliary Resources" on page 129 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 127 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 132 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 122 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 124 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 121, 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 121, 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 123.

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 121.

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 128 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 127.

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 125, 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 125 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 82) 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 120), 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 111 for details
	A_START	See the section "Progress Updating" on page 111 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 111 for details
	PROJECT	Activity is at highest level
	REMDUR	See the section "Progress Updating" on page 111 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 63 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

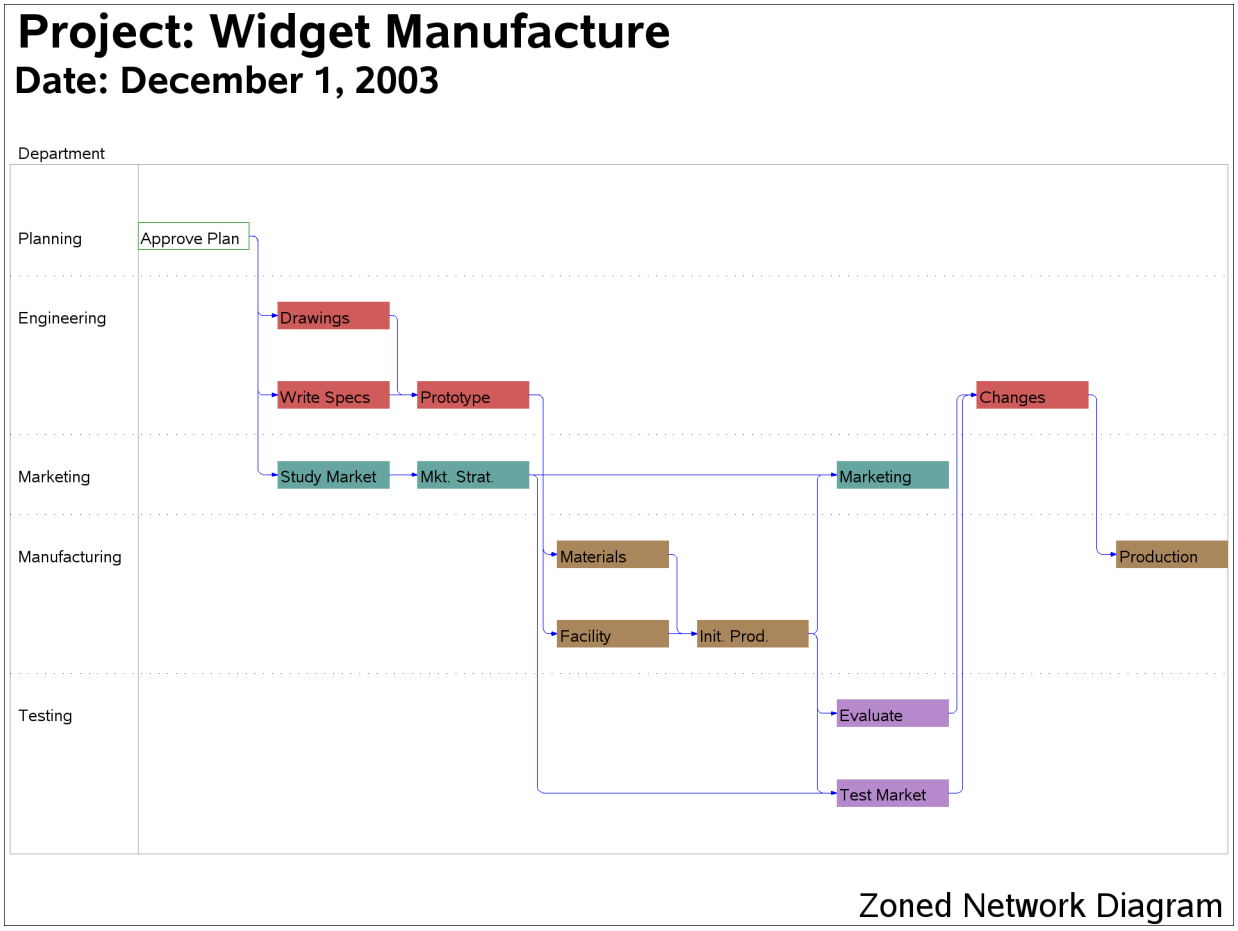
Table 4.12 Widget Manufacture: Precedence Information

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;
```

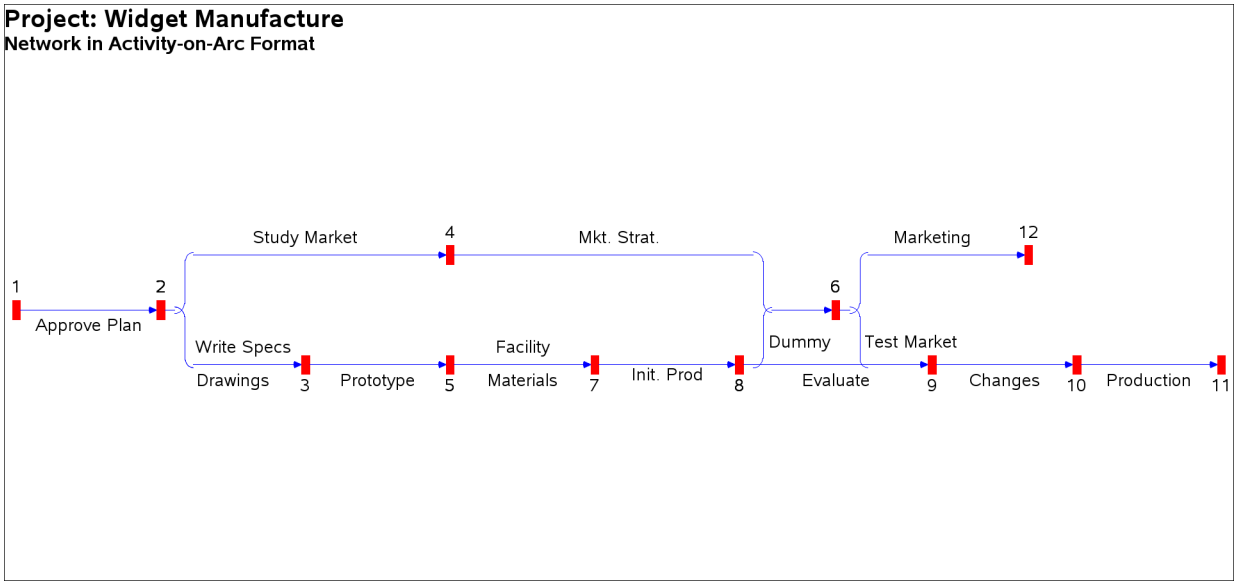
Output 4.1.3 Critical Path: Activity-On-Node Format

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

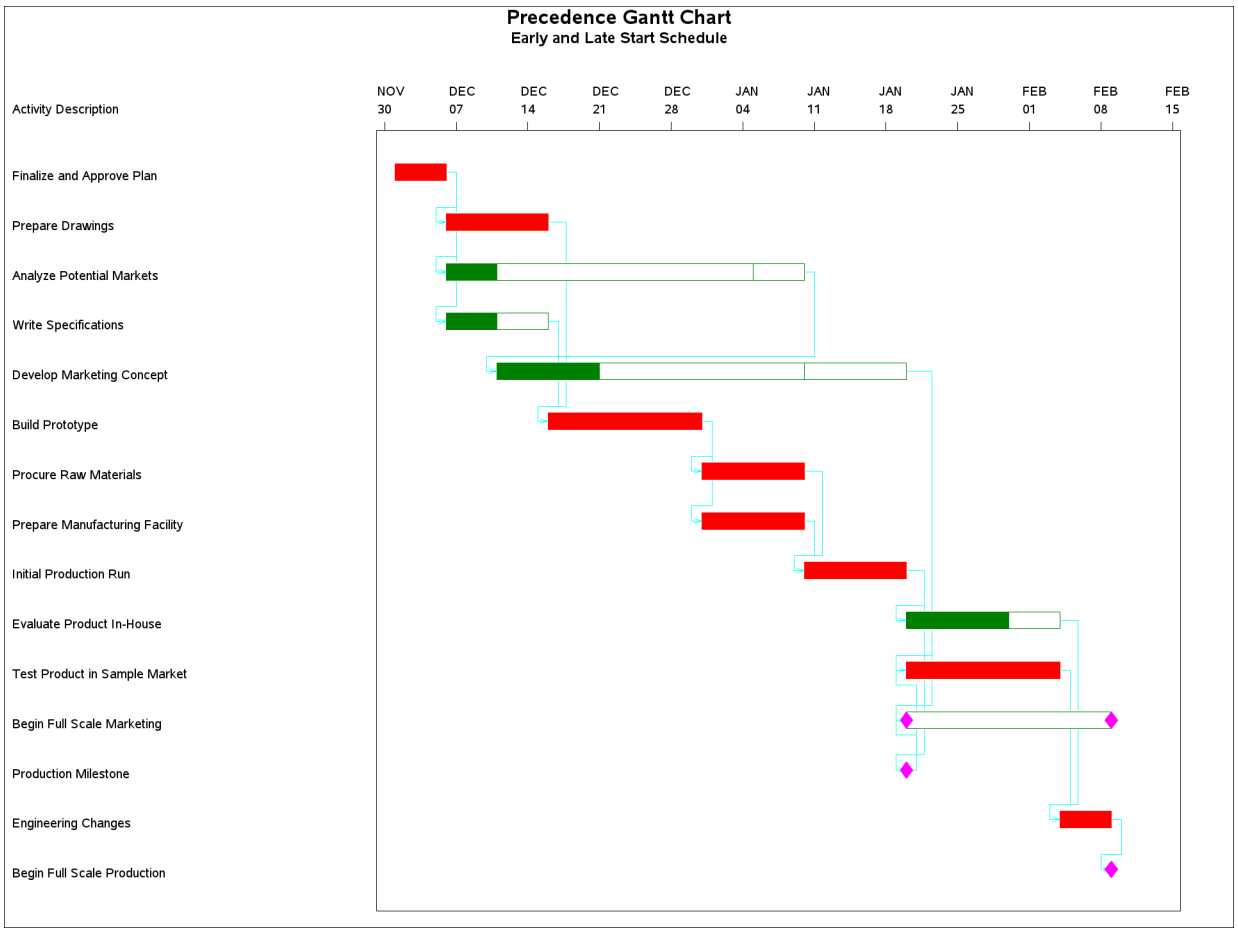
```
------------------------------------------------------------------------------
|                                                                            |
|                              December  2003                                |
|                                                                            |
|----------------------------------------------------------------------------|
|  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	.	.

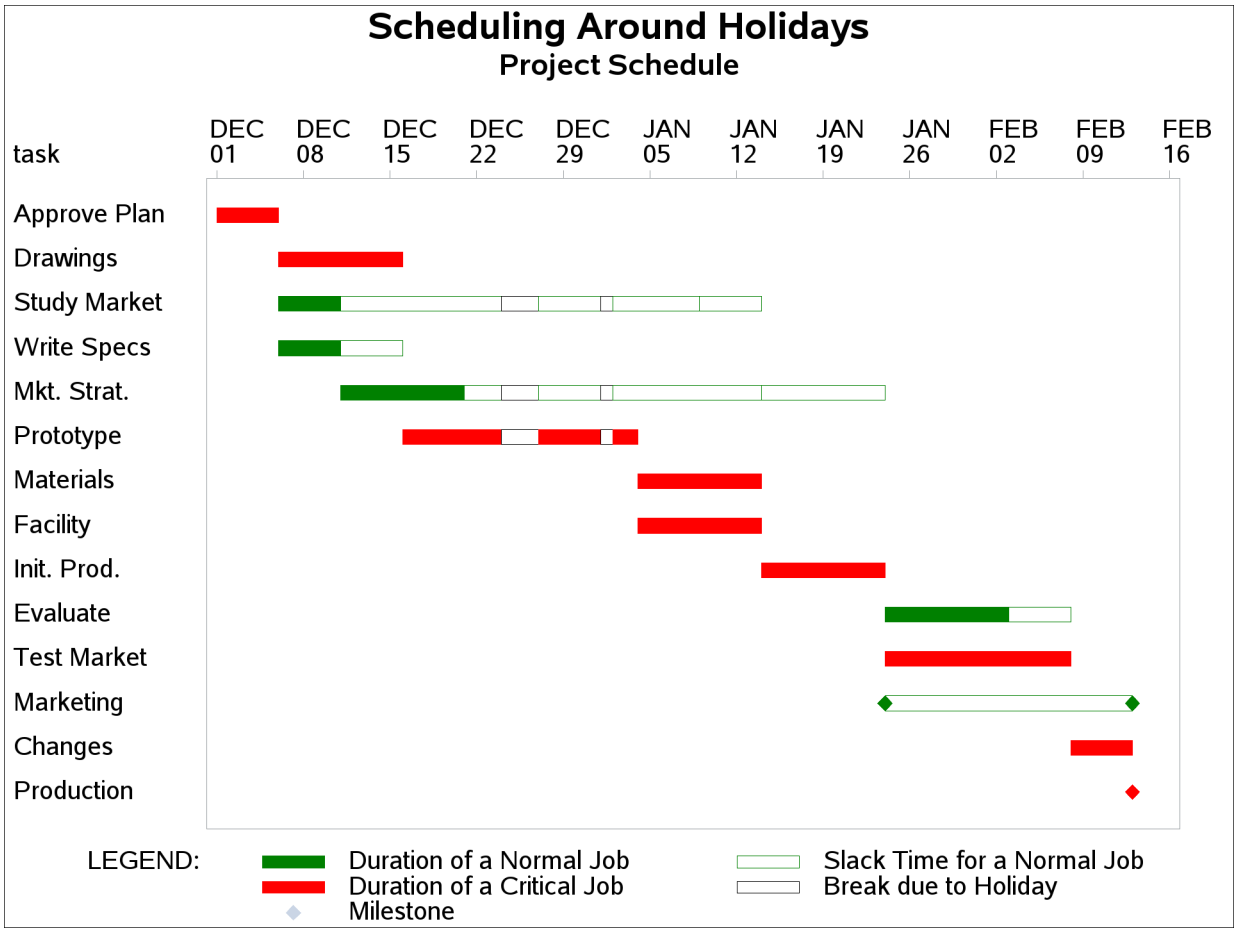
For example, the data set HOLIDAYS, displayed in Output 4.8.1 specifies two holidays, one for Christmas and the other for New Year's Day. The variable holiday specifies the start of each holiday. The variable holifin specifies the end of the Christmas holiday as 26Dec03. Alternately, the variable holidur 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;
```

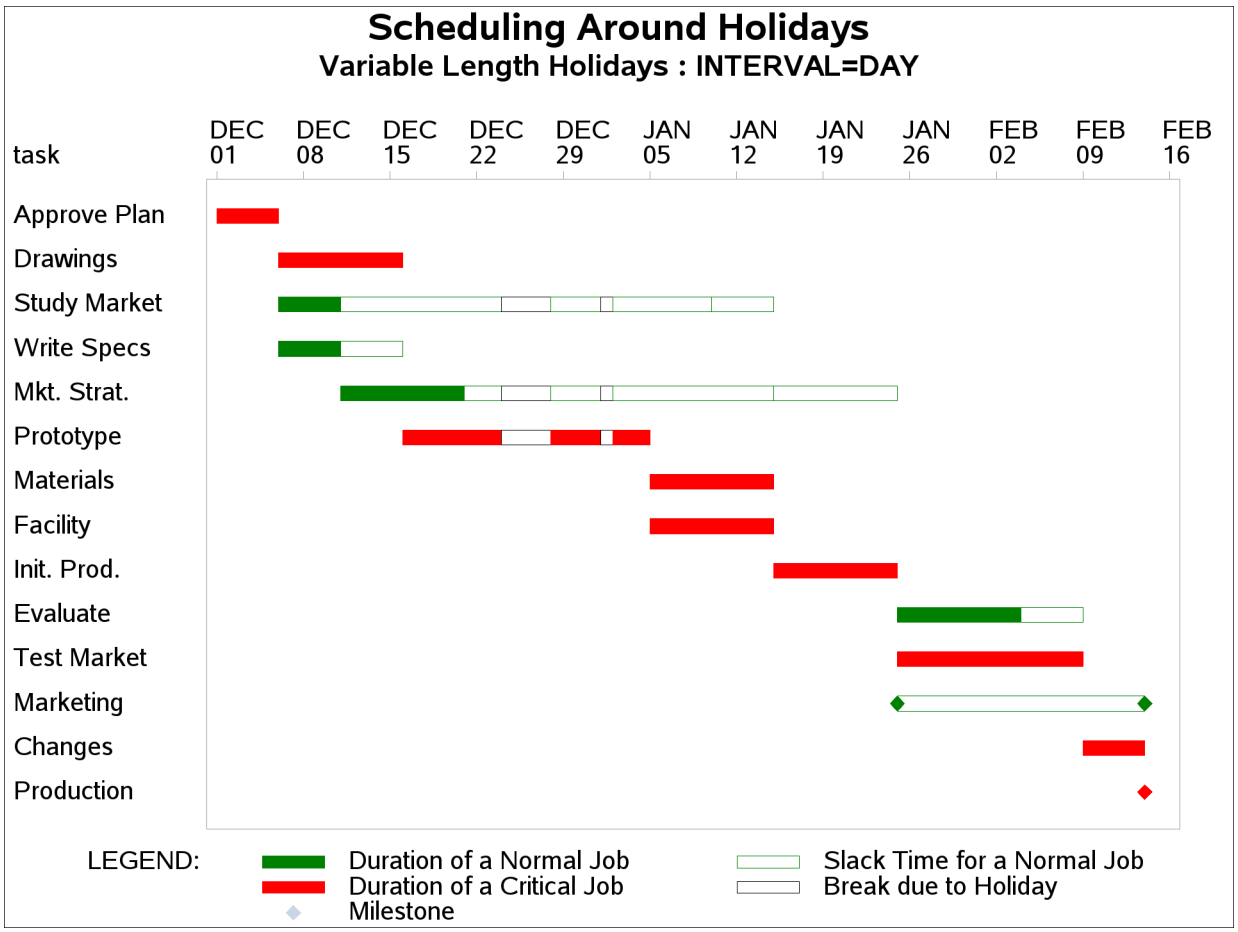
Output 4.8.2 Scheduling around Holidays: Project Schedule



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

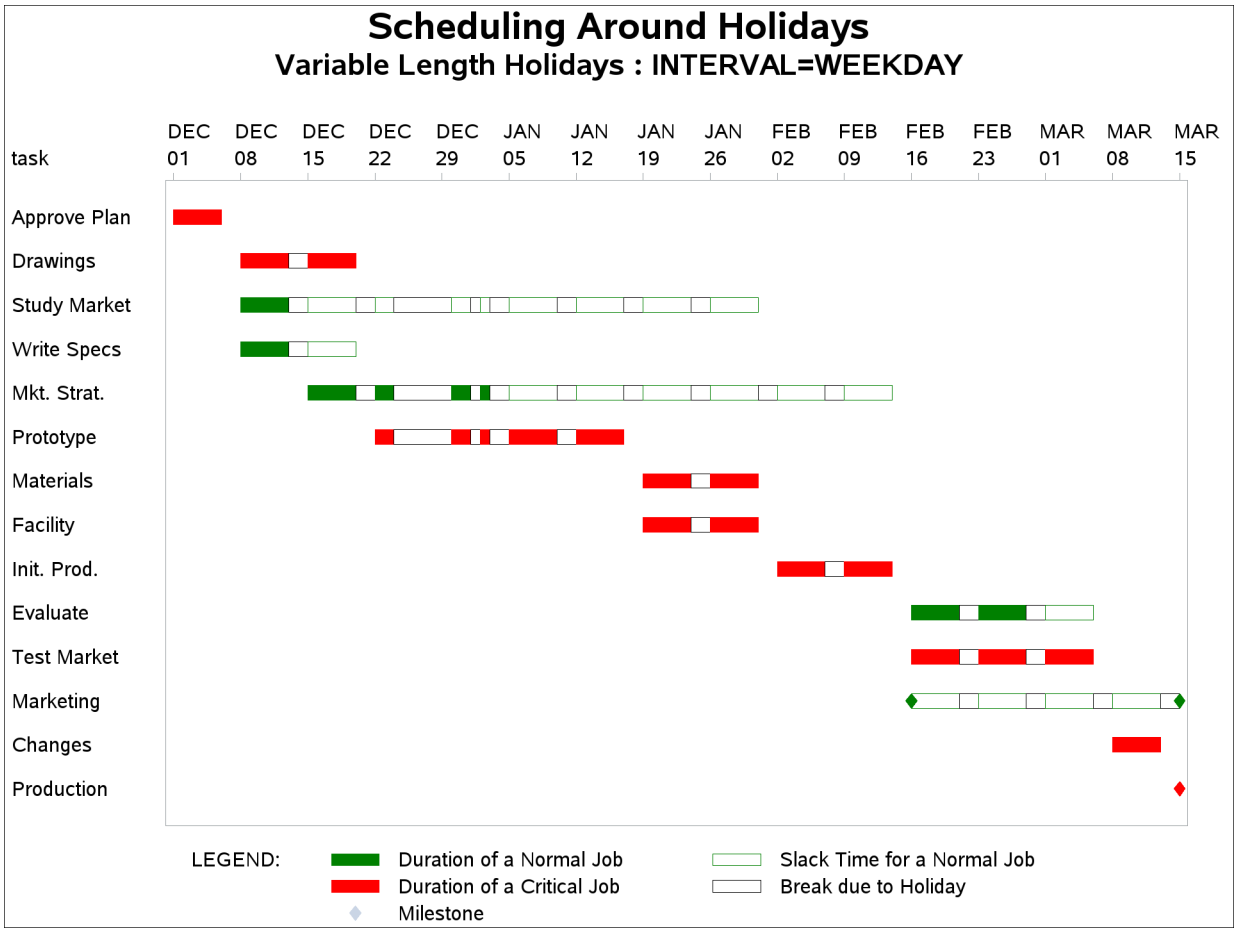


Next, suppose that work on the project is to be scheduled only on weekdays. The INTERVAL= option is set to WEEKDAY. Then, the value '4' specified for the variable holidur is interpreted as 4 weekdays. Thus, the holidays are on December 24, 25, 26, and 29, 2003, because December 27 and 28 (Saturday and 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;
```

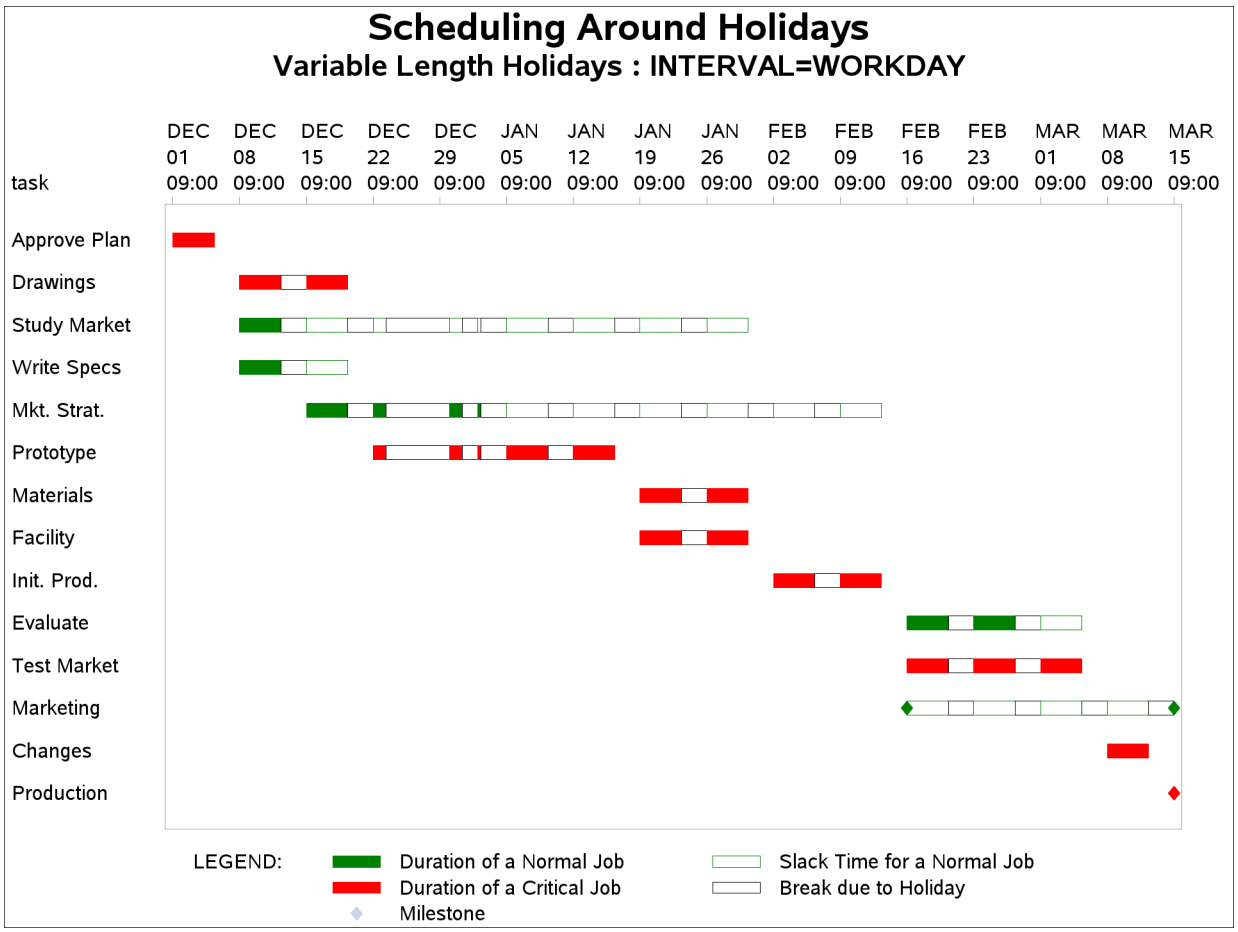
Output 4.8.4 Scheduling around Holidays: INTERVAL=WEEKDAY



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

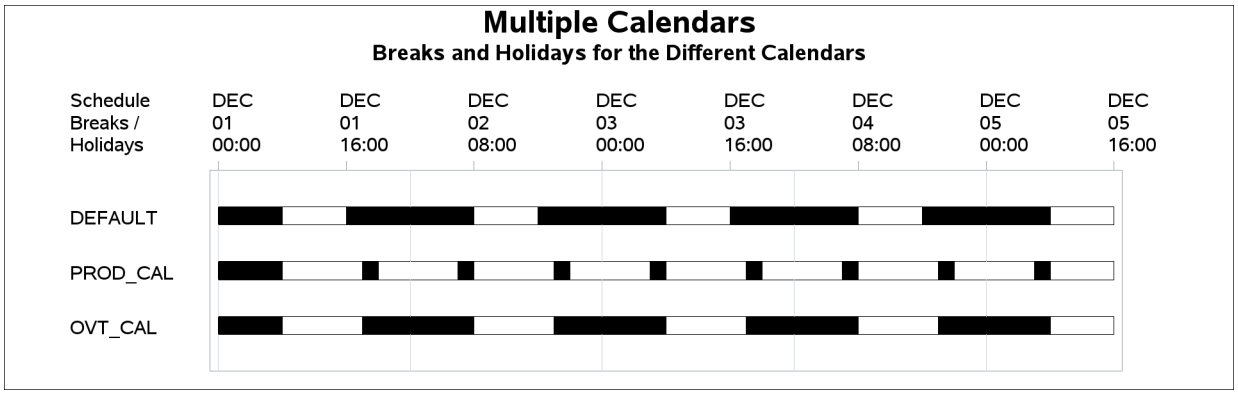
Obs	holiday	holifin	holidur
1	24DEC03	26DEC03	4
2	01JAN04	01JAN04	.

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.

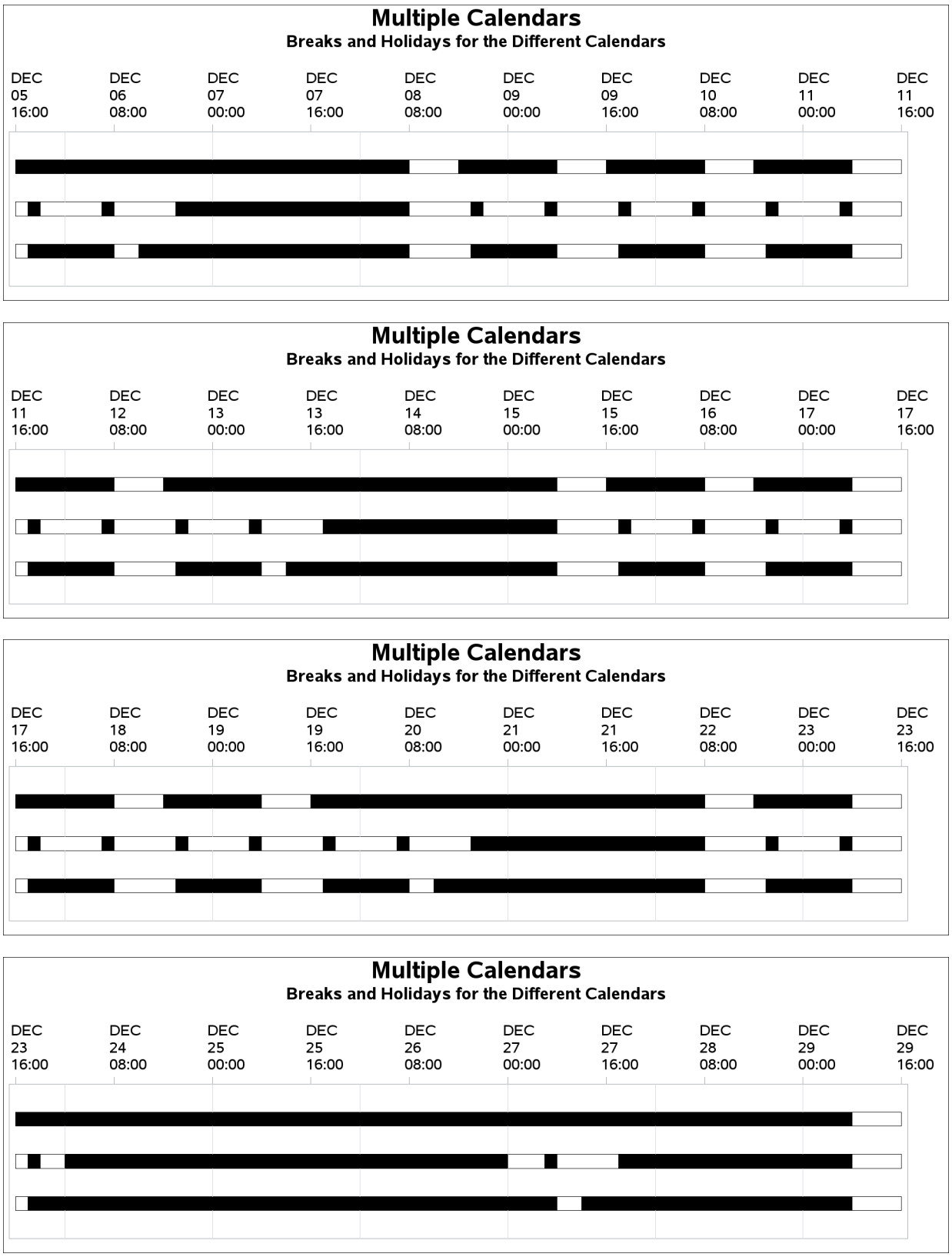
An easy way to visualize all the breaks and holidays for each calendar is to use a Gantt chart, plotting a bar for each calendar from the start of the project to January 4, 2004, with all the holiday and work shift specifications. The following program produces Output 4.10.3. Holidays and breaks are 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;
```

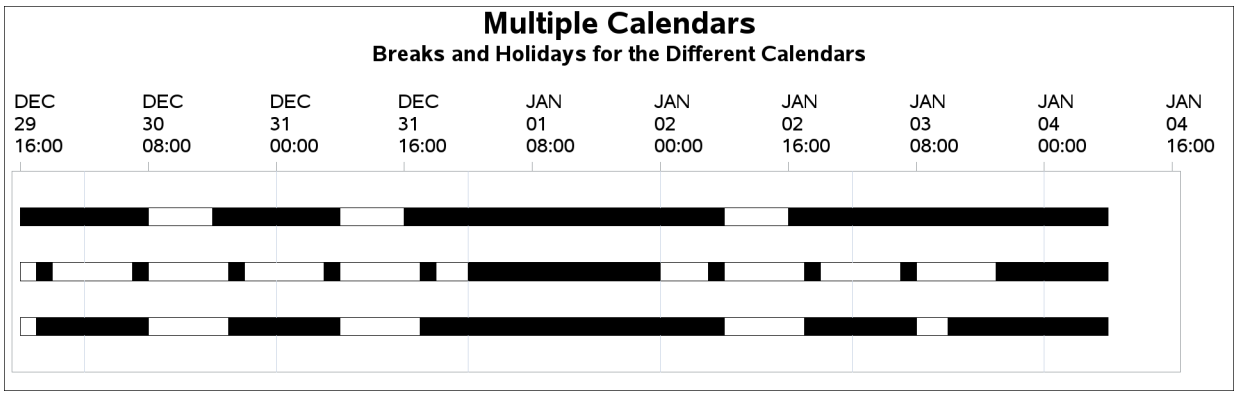
Output 4.10.3 Gantt Chart Showing Breaks and Holidays for Multiple Calendars



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	.	

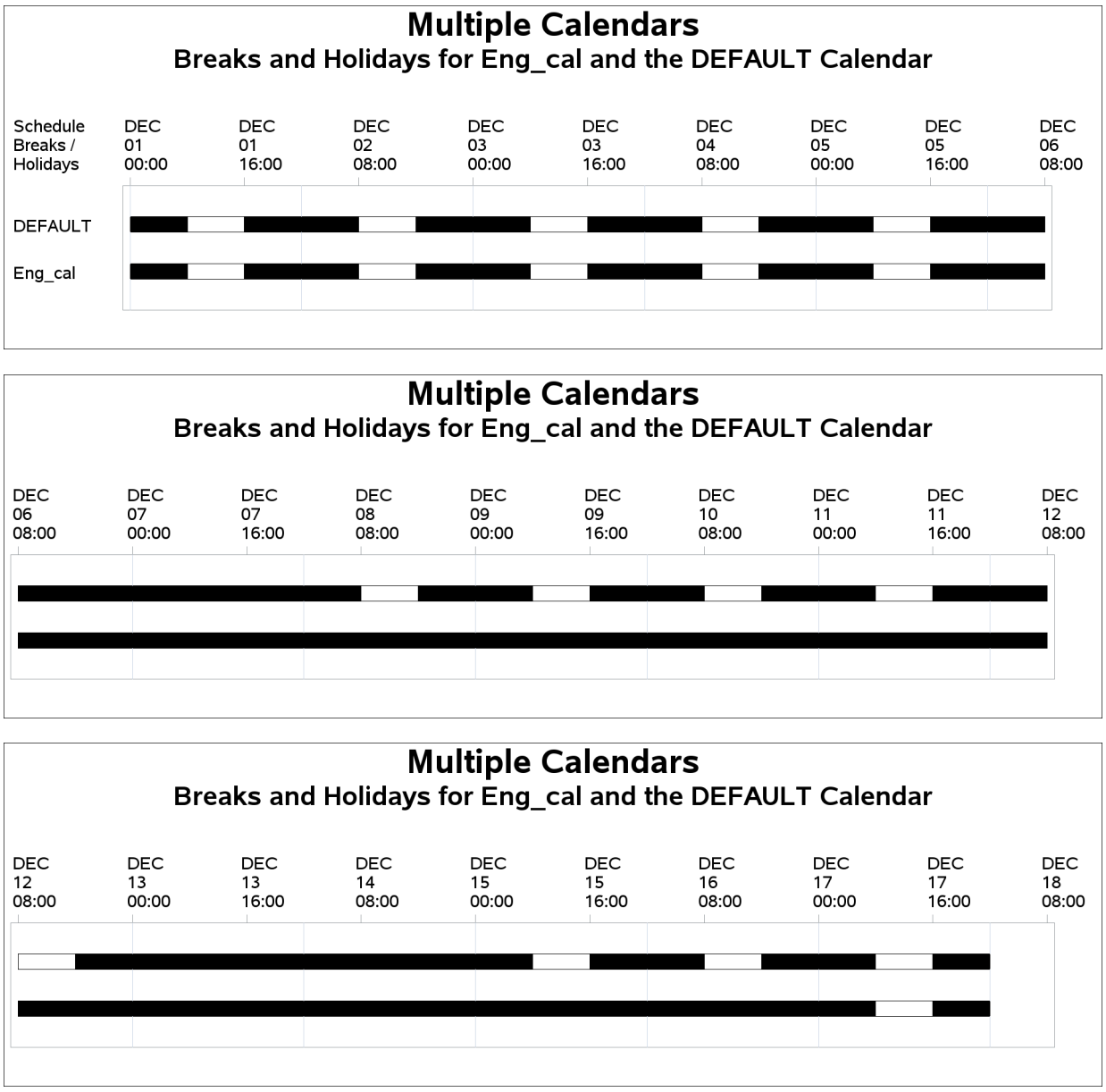
Multiple Calendars
Calendar Data Set

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';
```

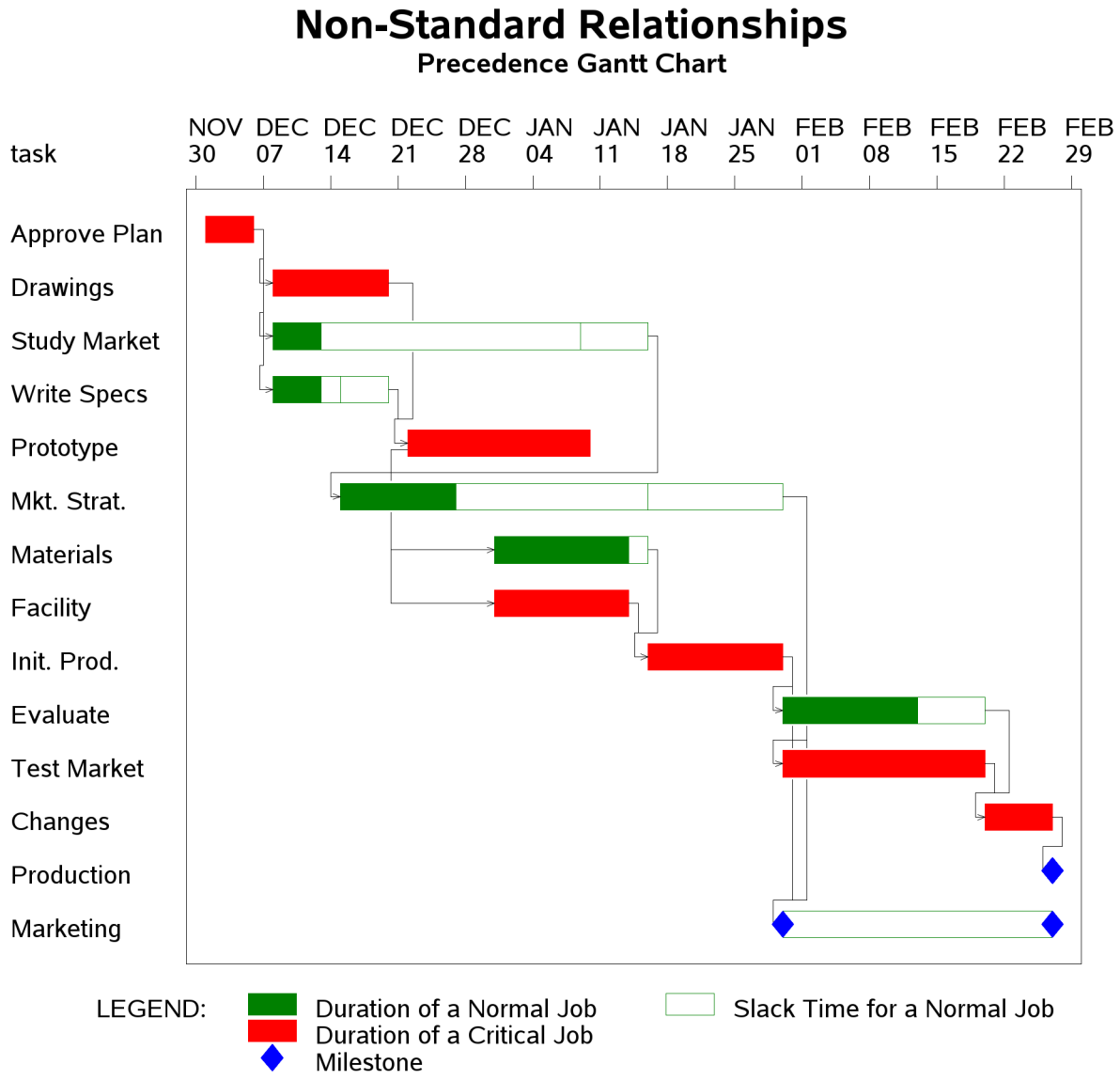
```
proc gantt graphics data=lagsched logic=widglag;
   chart / compress act=task succ=(succ) dur=days
           cprec=black cmile=blue
           caxis=black
           height=1.5 nojobnum
           dur=days increment=7 lag=(lagdur);
   id task;
   run;
```

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 LSS Eng	2 ESS Eng 2 LSS Eng	2 ESS Eng 2 LSS Eng	2 ESS Eng 2 LSS Eng	2 ESS Eng 2 LSS Eng
8	9	10	11	12
4 ESS Eng 1 LSS Eng	4 ESS Eng 1 LSS Eng	4 ESS Eng 1 LSS Eng	4 ESS Eng 1 LSS Eng	4 ESS Eng 1 LSS Eng
15	16	17	18	19
1 ESS Eng 3 LSS Eng	1 ESS Eng 3 LSS Eng	1 ESS Eng 3 LSS Eng	1 ESS Eng 3 LSS Eng	1 ESS Eng 3 LSS Eng
22	23	24	25 *Christmas*	26
4 ESS Eng 4 LSS Eng	4 ESS Eng 4 LSS Eng	4 ESS Eng 4 LSS Eng		4 ESS Eng 4 LSS Eng
29	30	31		
4 ESS Eng 4 LSS Eng	4 ESS Eng 4 LSS Eng	4 ESS Eng 4 LSS Eng		

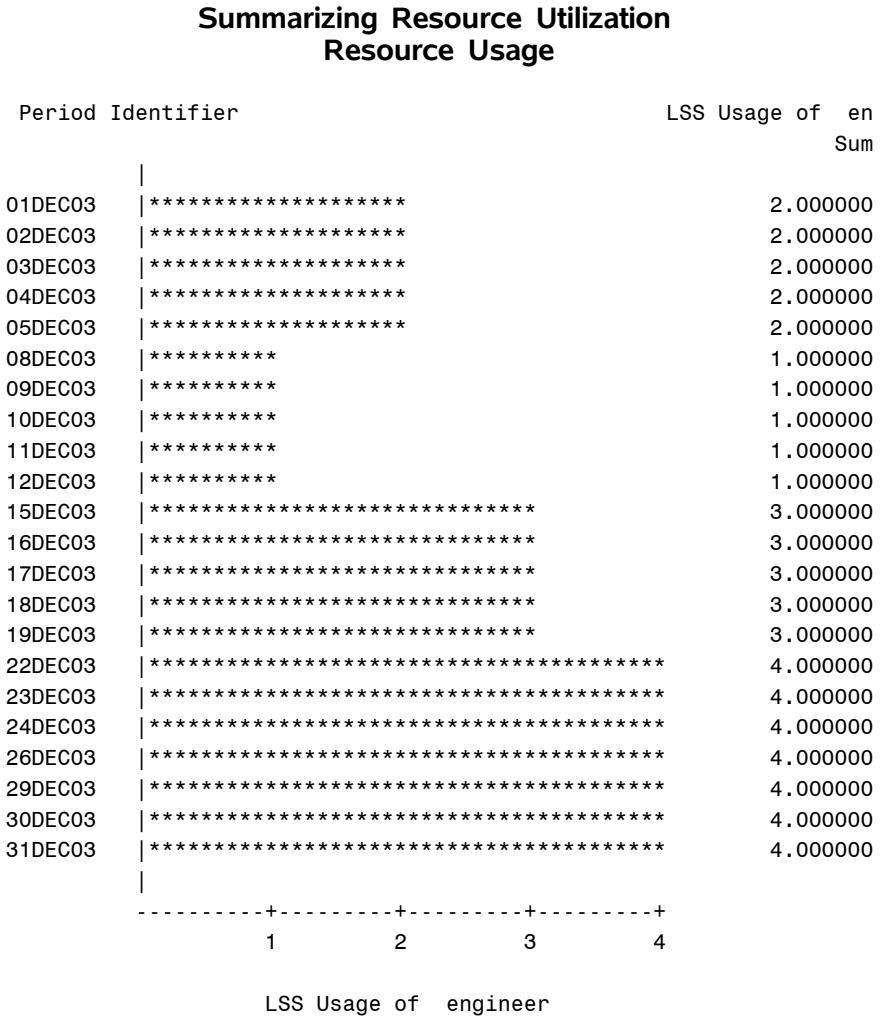
Legend
ESS Usage of engineer
LSS Usage of engineer

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



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_	Engineer	Engineer	Engineer	Engineer	Engineer Cost	Engineer Cost	Engineer Cost	Engineer Cost
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	Eengcost	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_	Engineer	Leeringineer	Reengineer	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 Engineer	Reng Engineer	Aeng Engineer	Eng cost	Leng cost	Reng cost	Aeng cost
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	Eng cost	Leng cost	Reng cost	Aeng cost
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	Engineer	Engineer	Engineer	Eng cost	Leng cost	Reng cost	Aeng cost
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_	Engineer	Engineer	Engineer	Engineer	Eng cost	Leng cost	Reng cost	Aeng cost
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	dedegg	prodedegg	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:

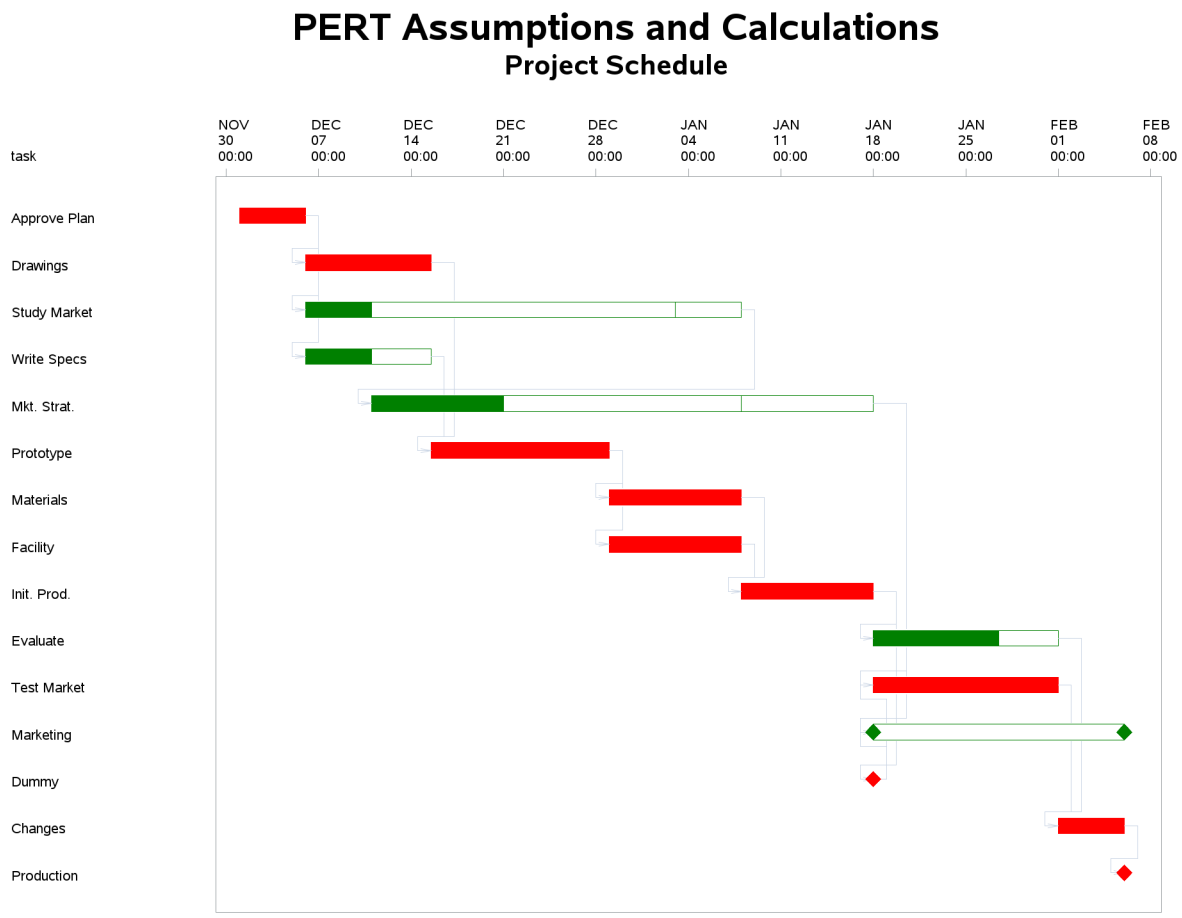
```
(tp + (4*tm) + to) / 6
```

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

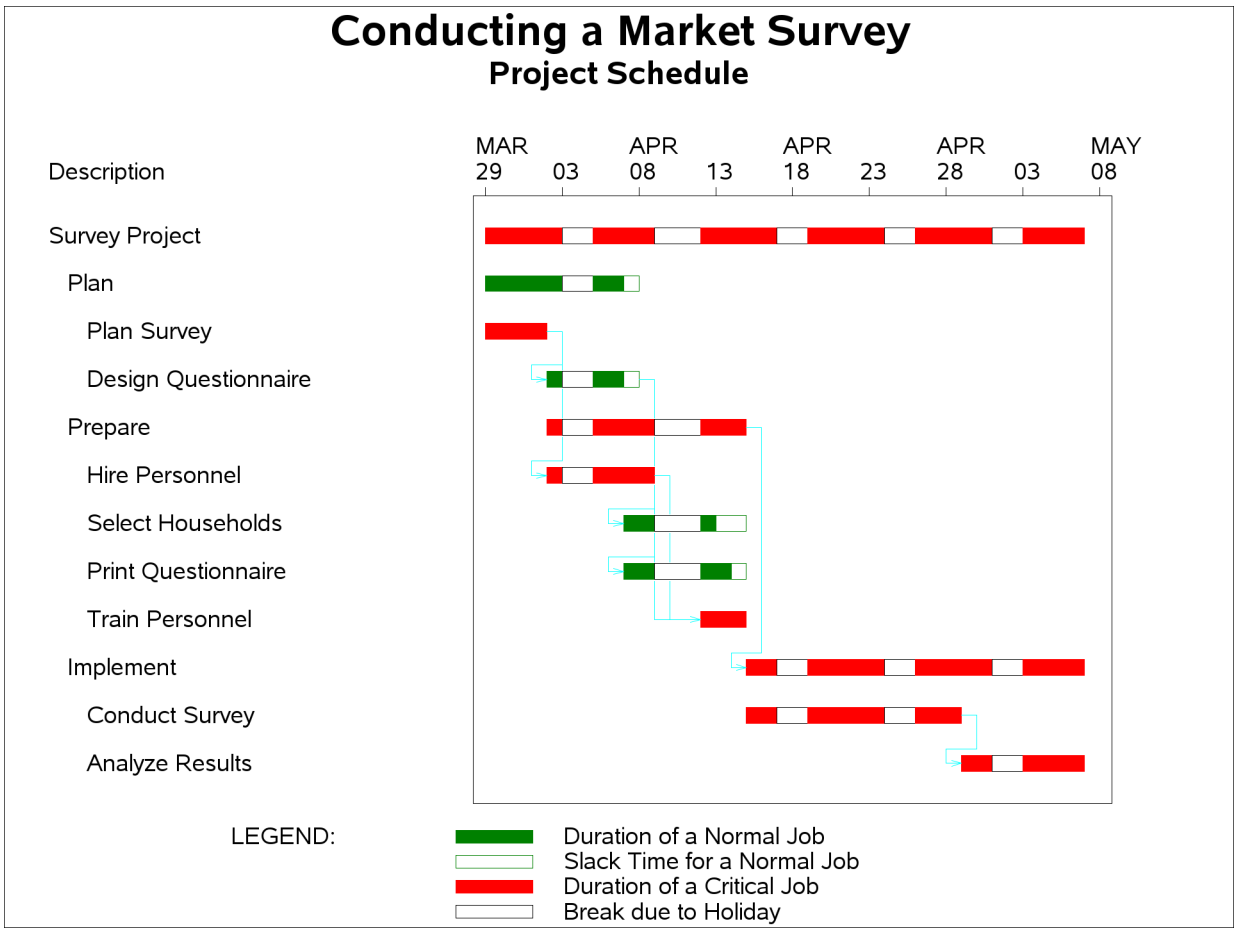
Obs	project	PROJ_DUR	PROJ_LEV	WBS_CODE	activity	succ1	succ2	succ3	duration
1		28	0	0	Survey				.
2	Survey	7	1	0.0	Plan				6
3	Plan	.	2	0.0.0	plan sur	hire per	design q		4
4	Plan	.	2	0.0.1	design q	trn per	select h	print q	3
5	Survey	8	1	0.1	Prepare	Implement			8
6	Prepare	.	2	0.1.0	hire per	trn per			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;
```

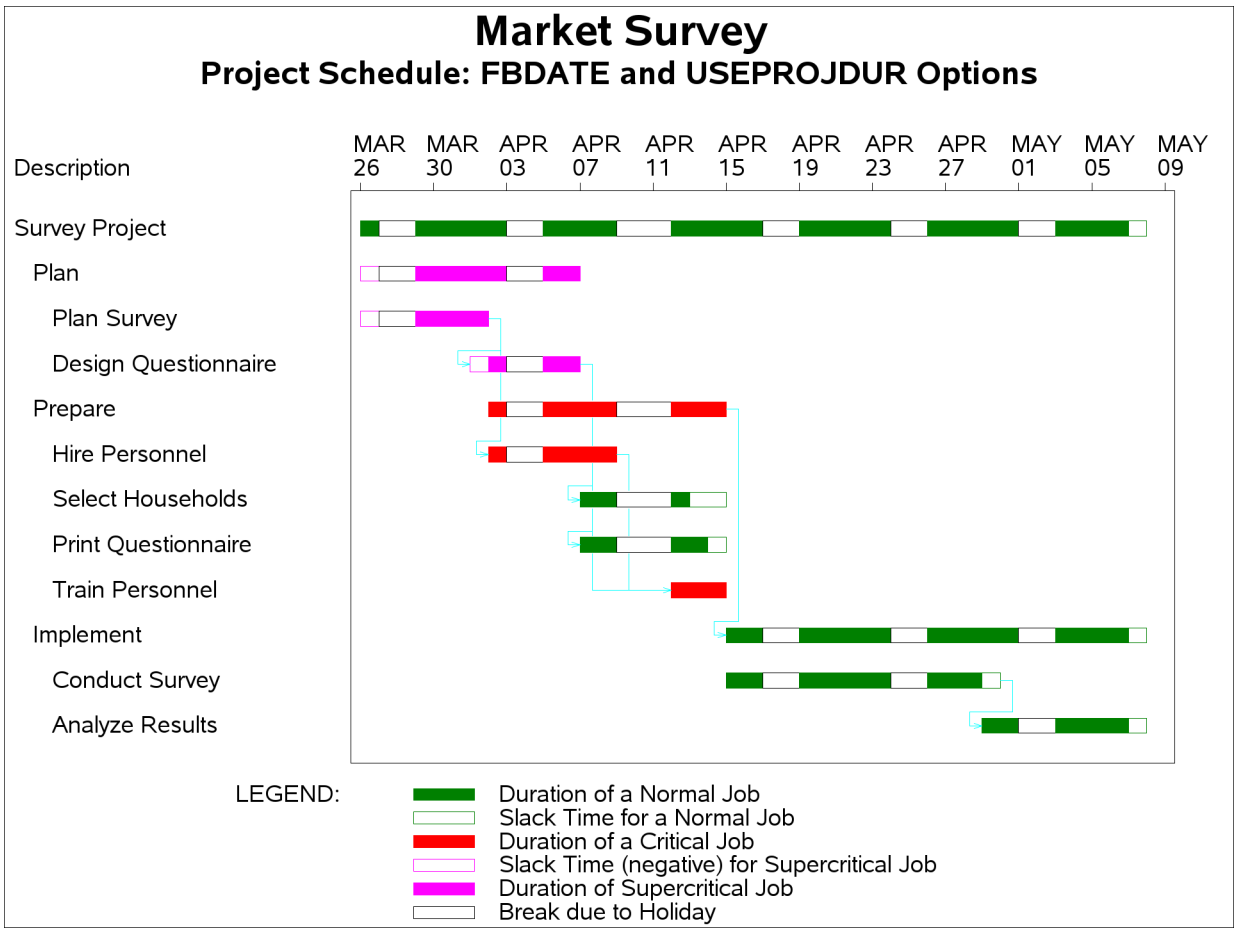
Output 4.23.8 Summary Schedule: FBDATE and USEPROJDUR Options

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

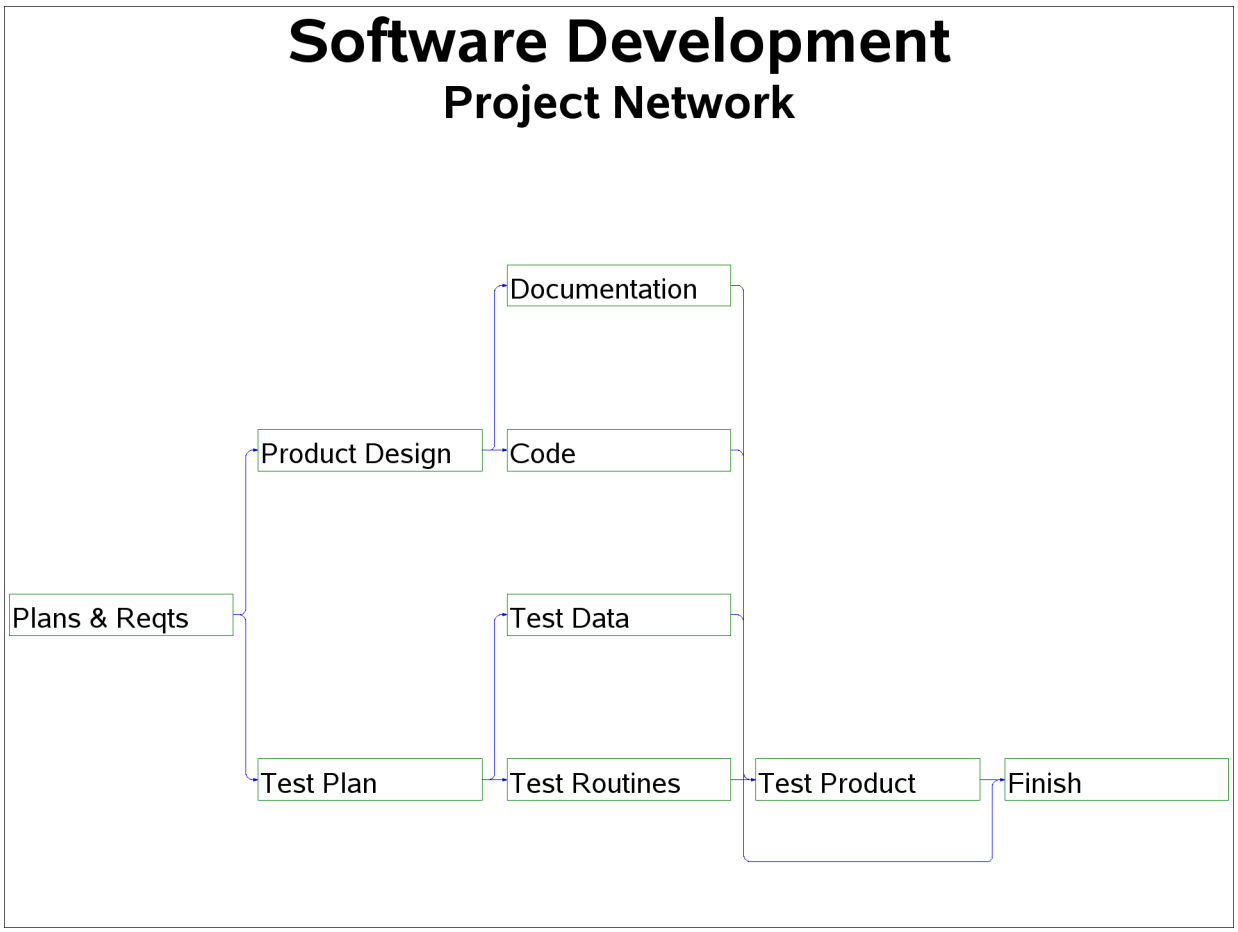
Example 4.24: Resource-Driven Durations and Resource Calendars

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 resssched=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

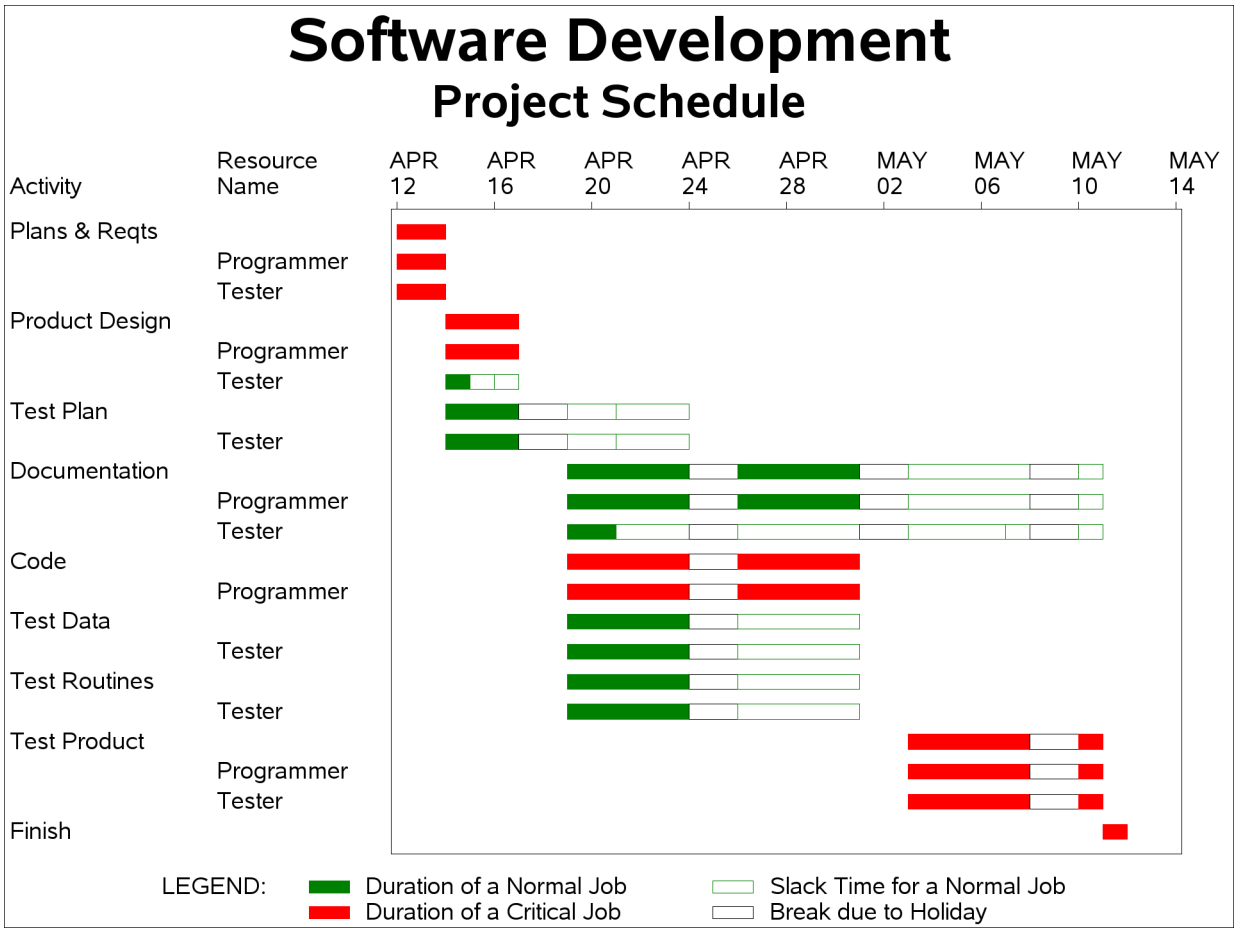
For each activity in the project, the Resource Schedule data set contains the schedule for the entire activity as well as the schedule for each resource used by the activity. The variable RESOURCE identifies the name of the resource to which the observation refers and has missing values for observations 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;
```

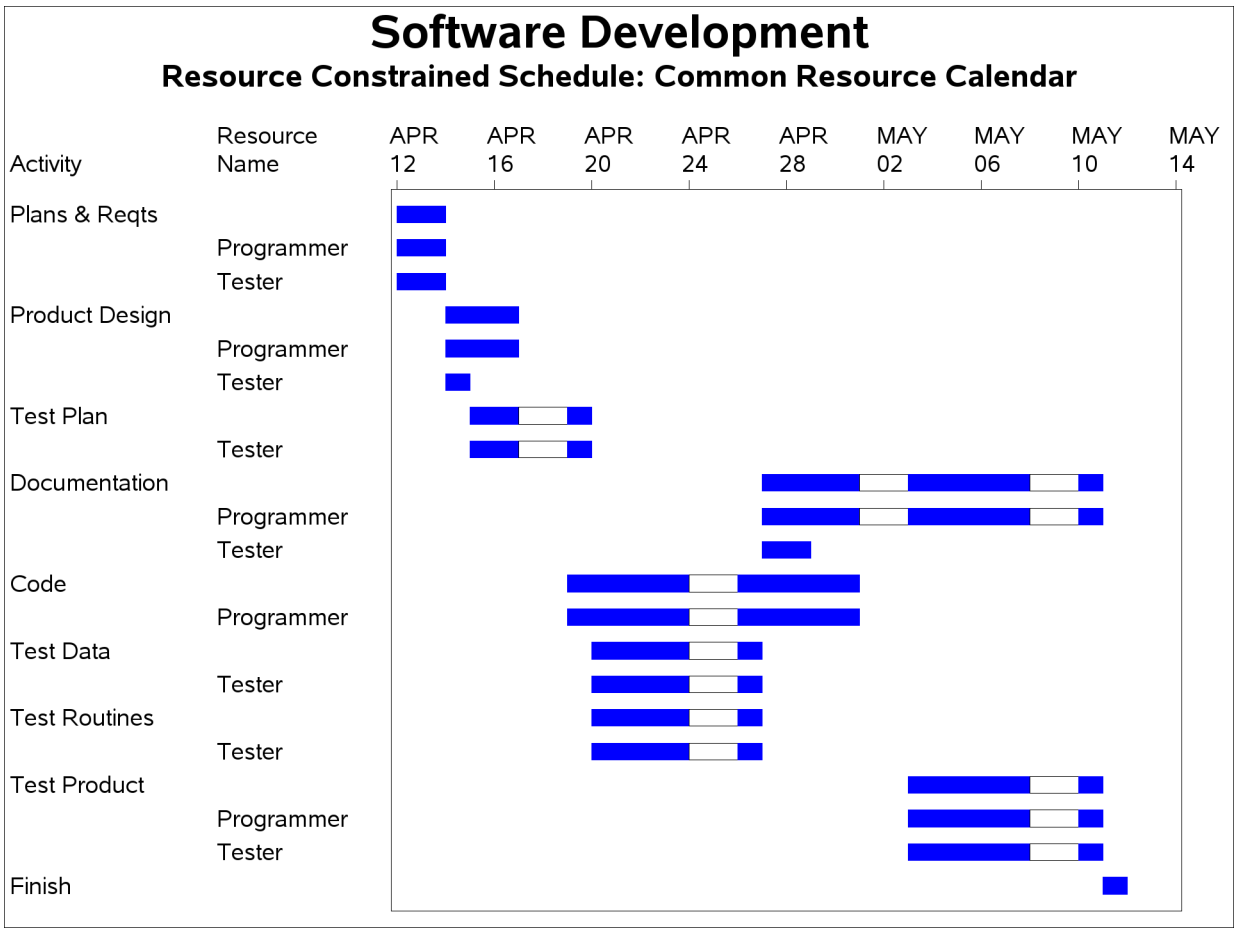
Output 4.24.7 Resource-Constrained Schedule: Common Calendar

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

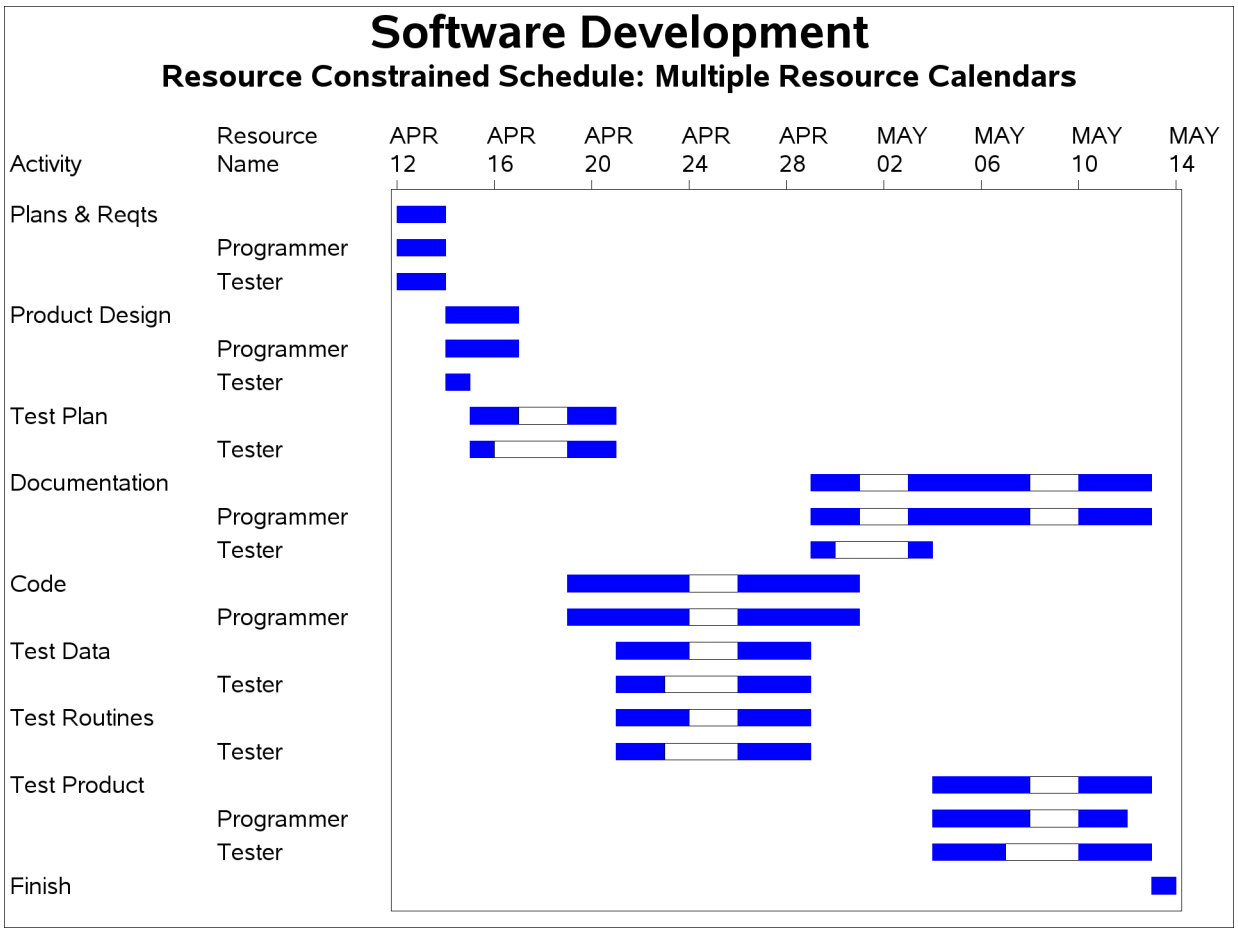
Resource Data Set RESIN2

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

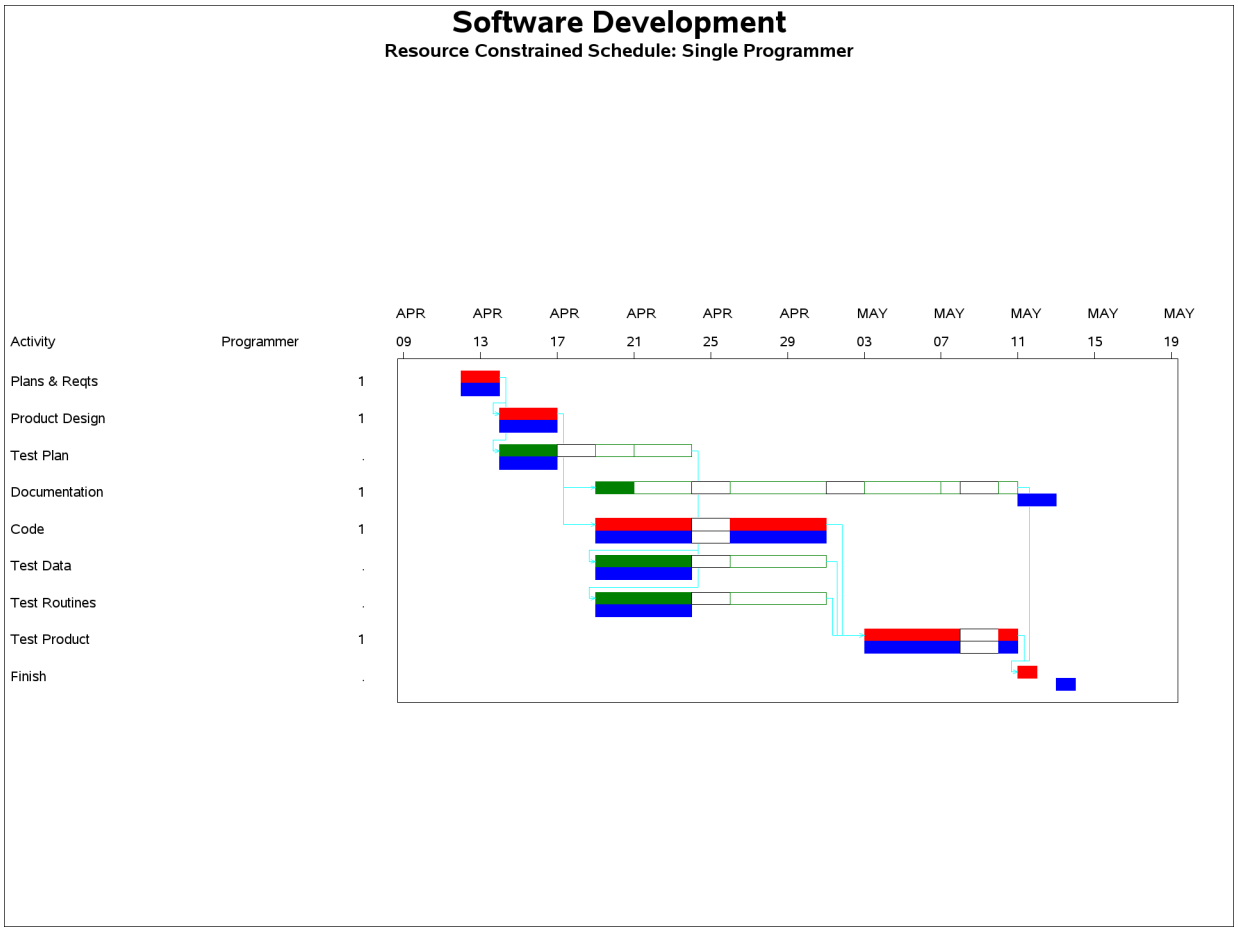
Activity	act	RESOURCE	DUR_TYPE	dur	mandays	R_RATE	S_START	S_FINISH
Plans & Reqts	1			2	.	.	12APR04	13APR04
Plans & Reqts	1	Programmer	FIXED	2	.	1	12APR04	13APR04
Product Design	2			3	.	.	14APR04	16APR04
Product Design	2	Programmer	RDRIVEN	3	3	1	14APR04	16APR04
Test Plan	3			3	.	.	14APR04	16APR04
Documentation	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 100 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 100, 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 SETFINISHMILESTONE 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

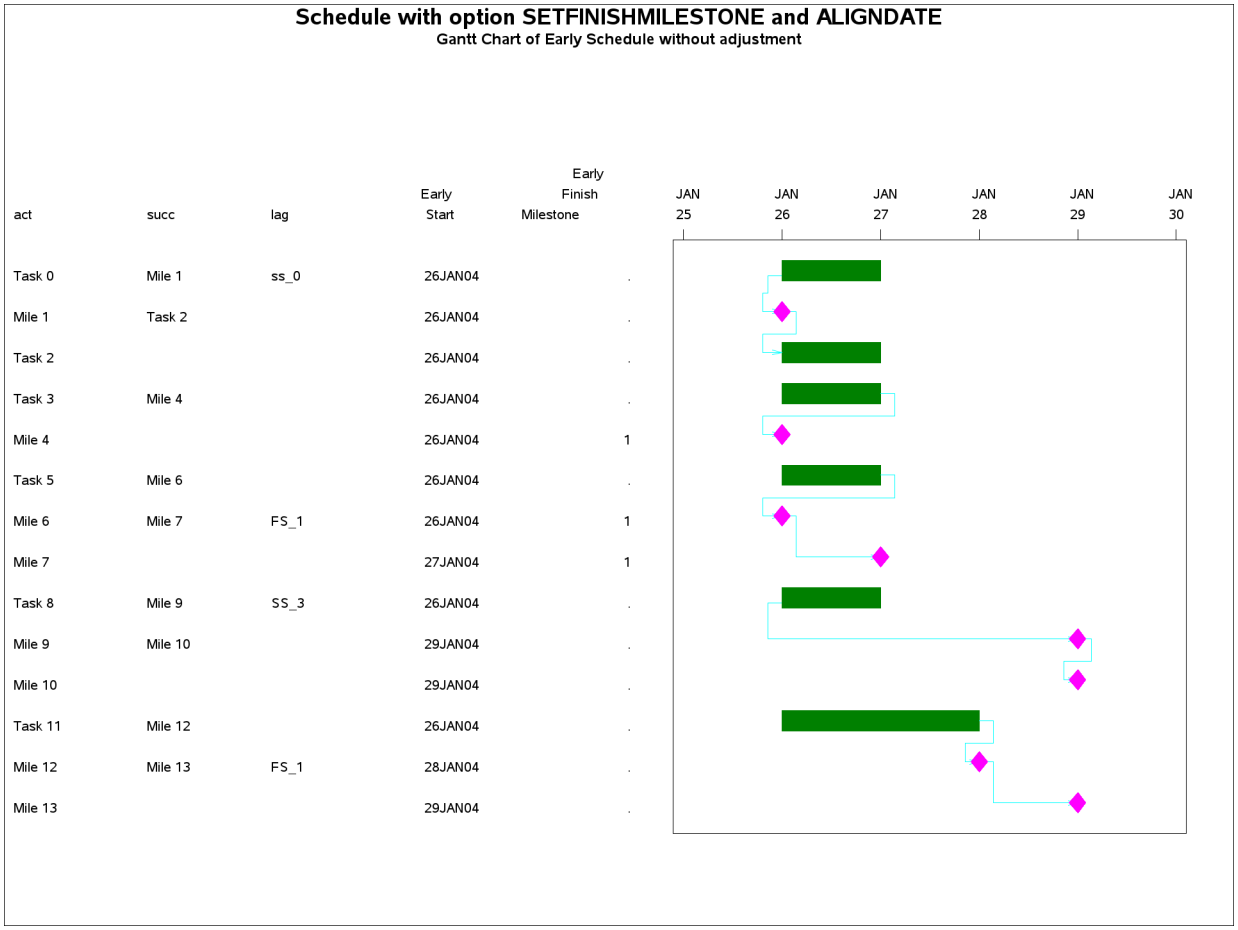
act	succ	lag	target	trgttype	miledate	E_START	E_FINISH	L_START	L_FINISH	EFINMILE	LFINMILE
Task 0	Mile 1	ss_0	26JAN04	SGE	.	26JAN04	26JAN04	28JAN04	28JAN04	.	.
Mile 1	Task 2		.		26JAN04	26JAN04	26JAN04	28JAN04	28JAN04	.	.
Task 2			.		.	26JAN04	26JAN04	28JAN04	28JAN04	.	.
Task 3	Mile 4		.		.	26JAN04	26JAN04	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.

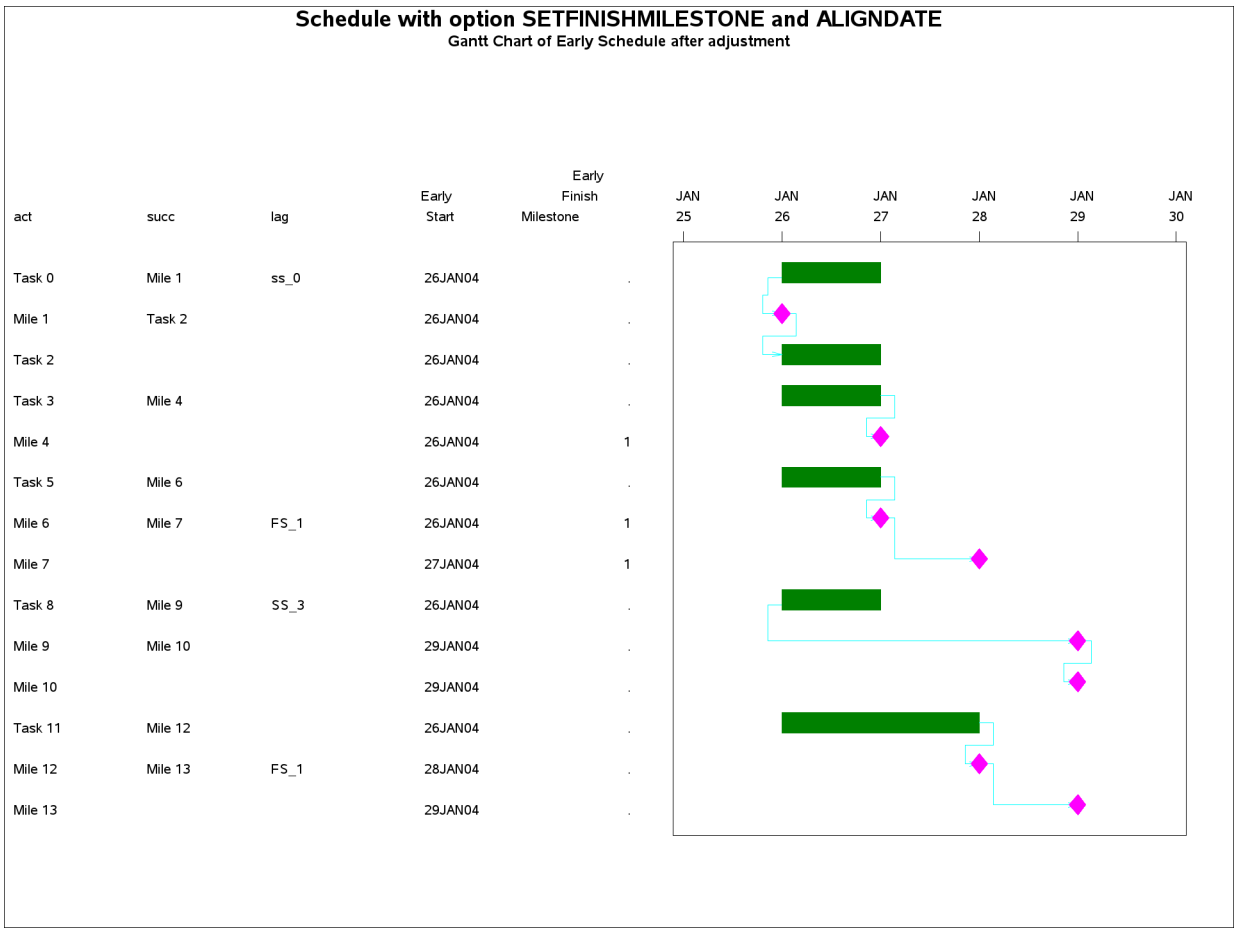
In this example, PROC GANTT is used to display the schedule produced in Output 4.28.4. Recall that the schedule is saved in the data set out2. First, PROC GANTT is invoked without any modifications to the schedule data set. The resulting Gantt chart is displayed in Output 4.28.5. The finish milestones (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



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

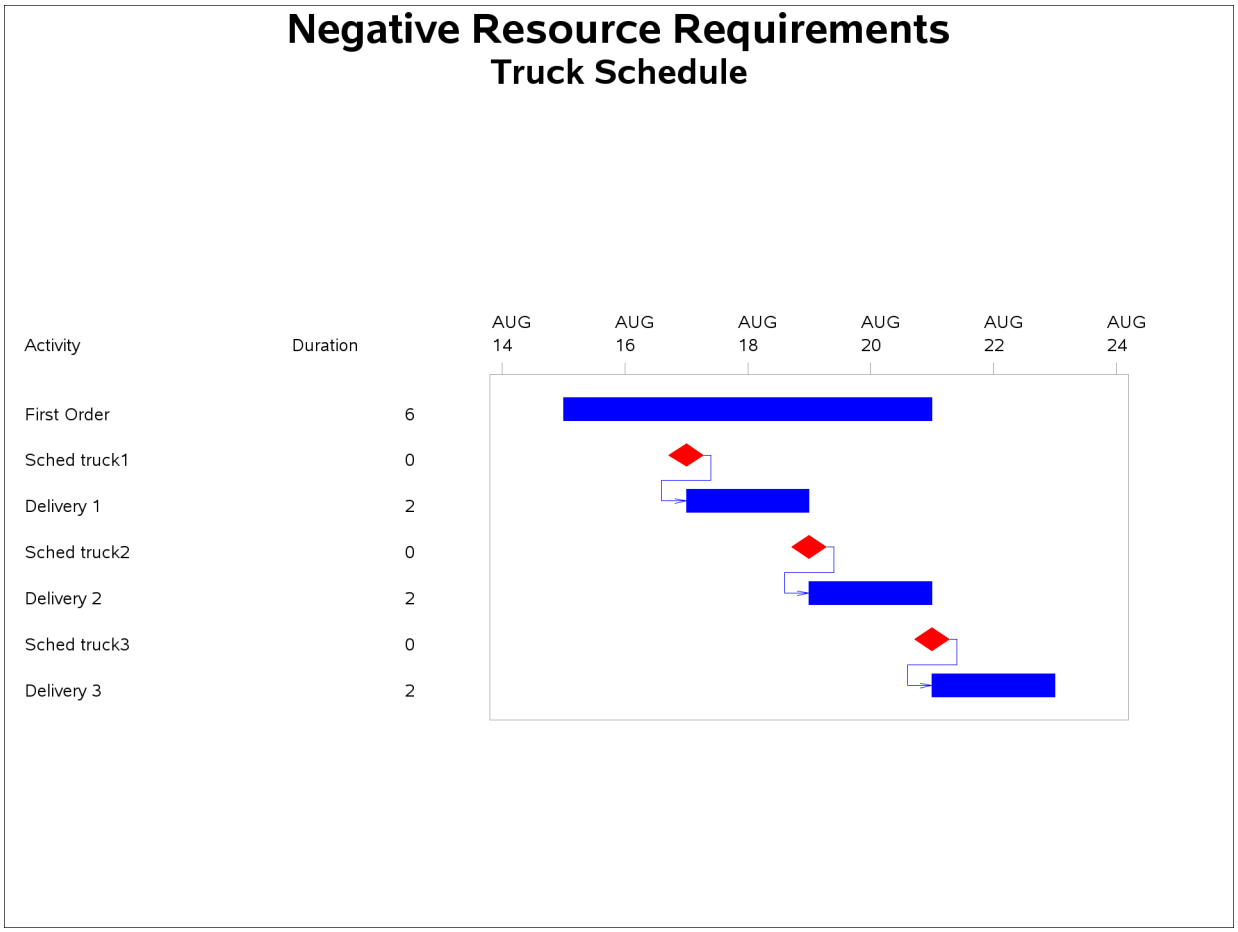
Obs	per	obstype	Mach1	numboxes	trucks
1	.	restype	1	2	1
2	15AUG04	reslevel	1	.	1

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

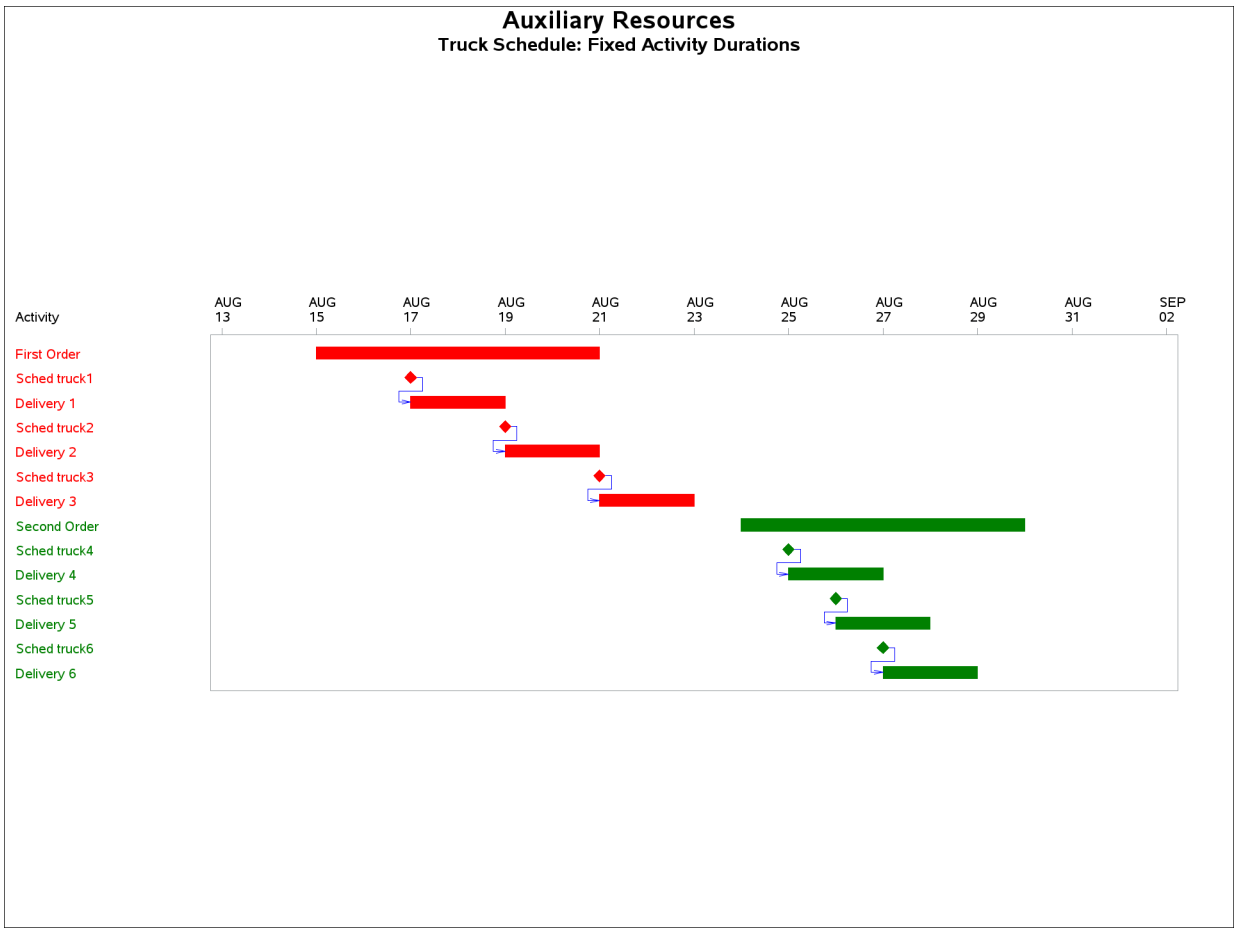
Auxiliary Resources
Resource Data Set TwoMachines

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	truck	_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

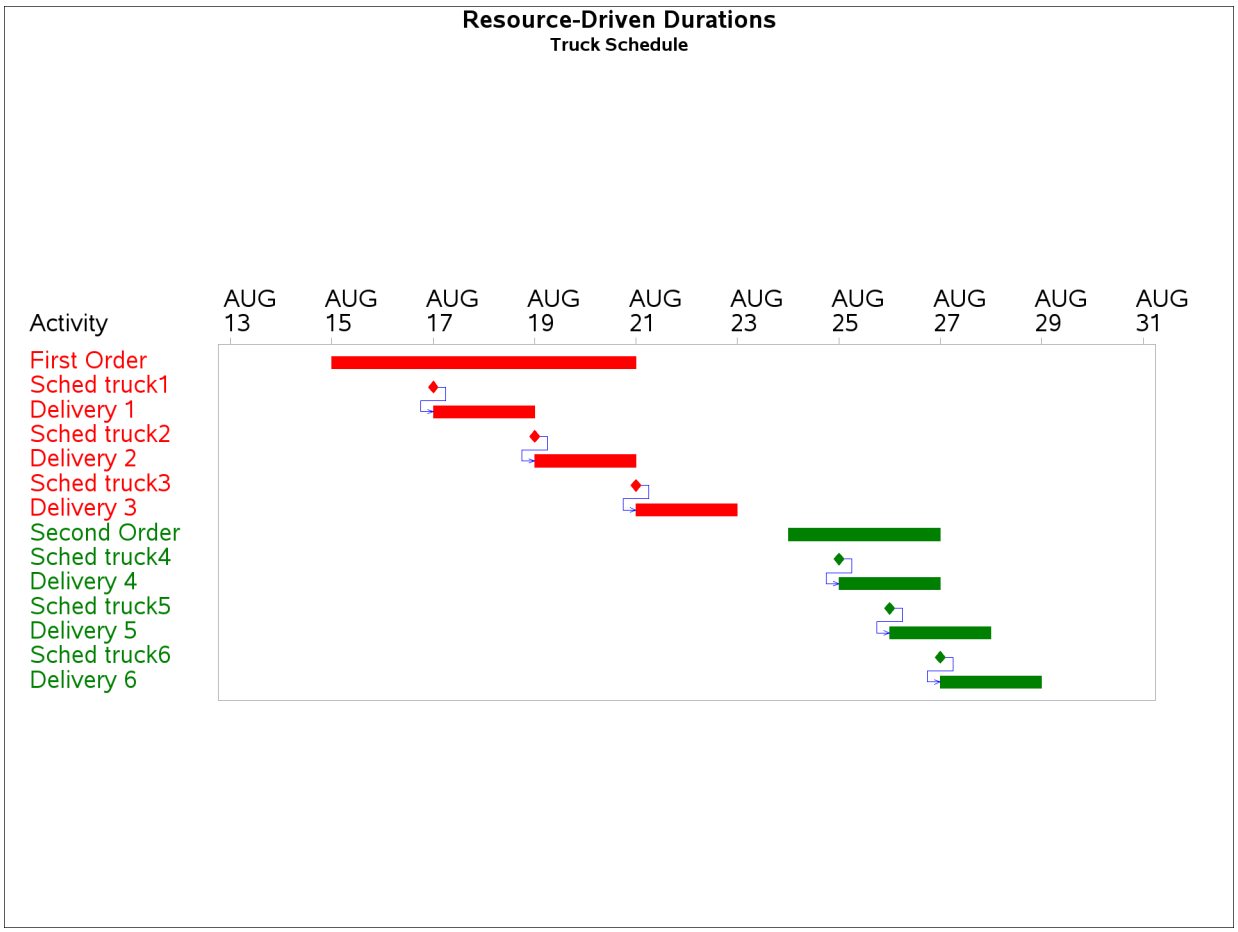
Resource-Driven Durations
Resource Data Set TwoMachinesRD

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.

Subject Index

ACTDELAY variable
  Activity data set (CPM), 83, 226
activity align types, 75
activity calendar, 77, 177
Activity data set
  CPM procedure, 57, 59, 68
  missing values, 139
  resource requirement specification (CPM), 120
  variables, 137
activity delay
  analysis, 84, 101, 203
  and supplementary resources, 123
  example (CPM), 217, 221, 222, 226
  specification, 83, 84
  wait until, 84
activity duration, *see* duration
activity information, *see* Activity data set
  additional variables, 80
Activity-on-Arc
  network diagram, 58
  specification, CPM procedure, 78, 93
Activity-on-Edge, *see* Activity-on-Arc
Activity-on-Node
  network diagram, 59
  specification, CPM procedure, 72, 92
Activity-on-Vertex, *see* Activity-on-Node
activity splitting
  at TIMENOW (CPM), 74, 114, 125
  CPM procedure, 86, 87, 91, 124, 125
  example (CPM), 228
  maximum number of segments, 86
  minimum duration of segment, 87
  option to allow default, 91
activity status
  setting in CPM procedure, 102
ACTIVITY variable
  Activity data set (CPM), 72
ACTIVITYPRTY variable
  Activity data set (CPM), 83
actual schedule
  CPM procedure, 72, 112–114
  example (CPM), 190
  finish times (CPM), 73
  start times (CPM), 73
add activity records
  CPM procedure, 67
A_DUR variable
  Schedule data set (CPM), 101, 112, 113, 192
A_FINISH variable
  Activity data set (CPM), 73, 96, 112
  Schedule data set (CPM), 101, 102, 112, 113, 192
AFINMILE variable
  Schedule data set (CPM), 101
ALIGNDATE variable
  Activity data set (CPM), 75, 96, 98, 99
alignment constraints
  CPM procedure, 75, 98, 99
ALIGNTYPE variable
  Activity data set (CPM), 75, 98, 99
alternate resources
  CPM procedure, 83, 87, 89, 125, 126
  example (CPM), 233
AOA, *see* Activity-on-Arc
AOE, *see* Activity-on-Arc
AON, *see* Activity-on-Node
AOV, *see* Activity-on-Node
arrow diagramming method, *see* Activity-on-Arc
A_START variable
  Activity data set (CPM), 73, 96, 112
  Schedule data set (CPM), 101, 102, 112, 113, 192
auxiliary resources
  CPM procedure, 129
  example (CPM), 292, 296

baseline schedule
  comparing, 76, 111, 226
  setting, 76, 77, 190
  specifying, 76, 77, 111
  updating, 77
B_FINISH variable
  Activity data set (CPM), 76
  Schedule data set (CPM), 77, 190
break and shift information
  discussion, CPM procedure, 103–107, 111
B_START variable
  Activity data set (CPM), 76
  Schedule data set (CPM), 77, 190

CAL variable, *see* CALID variable
  Activity data set (CPM), 77
  Calendar data set (CPM), 77
  Holiday data set (CPM), 77
  Resource Schedule data set (CPM), 83
Calendar data set
  calendars example (CPM), 105
  CPM procedure, 57, 67, 103, 105, 170, 174
  missing values, 139

variables, 137
calendar information, *see* Calendar data set
calendars, *see* Calendar data set, *see* Holiday data set, *see* holidays, *see* multiple calendars and holidays, *see* Workday data set
associated with activity, 61
default, CPM procedure, 104
discussion, 103–107, 111
in Usage data set, 130
length of workday (CPM), 68, 96, 104, 105
multiple calendars, 103–107, 111
start of workday (CPM), 68, 96, 104
work shifts, 104
work unit specification (CPM), 69
CALID variable
Activity data set (CPM), 77, 103, 104
Calendar data set (CPM), 77, 103–106
example (CPM), 177, 181
Holiday data set (CPM), 77, 103, 104, 106, 107
common working calendar, 89
comparison of schedules, 76
computer resource requirements
CPM procedure, 70, 141
consumable resource, 117
CPM, *see* critical path method
CPM examples
activity splitting, 228
Activity-on-Arc format, 148
Activity-on-Node format, 144
alternate resources, 233
analyzing resource delay, 201
auxiliary resources, 292, 296
basic project schedule, 142
changing length of workday, 161
changing start of workday, 161
course scheduling, 244
finish milestone, 281
incorporating actual schedule, 190
infeasibility diagnostics, 214
meeting project deadlines, 151
multiproject scheduling, 248
negative resource requirements, 289, 292, 296
nonstandard precedence constraints, 183
PERT analysis, 241
resource calendars, 258, 296
resource-driven durations, 258, 296
resource-constrained scheduling, 201
saving a target schedule, 190
scheduling around holidays, 163
scheduling courses, 244
scheduling only on weekdays, 157
scheduling over nonstandard day and week, 169, 174
setting activity delay, 221
setting project finish date, 151
setting project start date, 145
substitutable resources, 233
summarizing resources used by project, 195
supplementary resources, 210
time-constrained scheduling, 188
TIMENOW option, 190
use of PROC CALENDAR to print schedule, 153
CPM procedure
Activity data set, 68
Activity-on-Arc, 58
Activity-on-Node, 59
actual schedule, 72
add activity records, 67
alternate resources, 125
auxiliary resources, 129
baseline schedules, 111
Calendar data set, 67, 103, 105
computer resource requirements, 70, 141
default calendar, 104
definitions of Schedule data set variables, 101–103
details, 94
duration specification, 96
finish milestone, 100, 101
float times, 95
formatting details, 141
functional summary, 63
Holiday data set, 69, 103, 106, 107
input data sets, 137
missing values, treatment of, 139
multiple alternates, 127
multiple calendars, 103–107, 111
multiproject scheduling, 133–136
negative resource requirements, 121
options classified by function, 63
output data sets, 101, 129, 133
overview, 57
precedence relationships, 97, 98
progress updating, 111–114
progress variables, 72, 112, 113
random activity durations, 243
resource allocation, 115, 116, 118–127, 130–133
Resource data set, 70, 116
resource-driven durations, 128
Resource Schedule data set, 71
resource usage, 131
SAS date, time, and datetime values, 68, 96, 97, 107, 141
Schedule data set, 70, 101–103
scheduling subject to precedence constraints, 95, 96
serial-parallel scheduling method, 121, 122
specifying resource requirements, 120, 121

syntax skeleton, 63
table of syntax elements, 63
target schedules, 111
time-constrained scheduling, 98, 99
Usage data set, 71
variables, 101, 137
Workday data set, 72, 103, 104
CPM procedure
macro variable _ORCPM_, 136
CPU requirement, *see* computer resource requirements
critical activities
CPM procedure, 60, 61, 95, 99, 103
critical path, 95, 103
critical path method, 58
critical path method (CPM), 60
cumulative resource usage, 84
current time, *see* TIMENOW
cycles
definition, CPM procedure, 96

data storage requirements, *see* computer resource requirements
day
length of, 68, 96, 104, 105, 162
start of, 68, 96, 104, 105, 162
deadlines
finish-before date, 68
default calendar
CPM procedure, 77, 104, 177
delay diagnostics, *see* activity delay
DELAY_R variable
Schedule data set (CPM), 84, 101, 203
delete duplicate observations, 68
diagnose resource infeasibilities, *see* infeasibility diagnostics
D_LENGTH variable
Calendar data set (CPM), 105
duplicate observations
deleting (CPM), 68
DUR variable, *see* DURATION variable
Resource Schedule data set (CPM), 133
duration
calculated, 78
estimates of, 241
multiplier, CPM procedure, 69
resource-driven, 92, 101, 114
specification, CPM procedure, 78, 96
units, CPM procedure, 61, 69, 96, 157
DURATION variable
Activity data set (CPM), 78, 114, 115
DUR_TYPE variable
Resource Schedule data set (CPM), 133

early start schedule computation, *see* schedule computation
E_FINISH variable
Schedule data set (CPM), 61, 85, 95, 101, 102
EFINMILE variable
Schedule data set (CPM), 101
errors
CPM procedure, 136, 139
ES_ASC variable
Schedule data set (CPM), 80
ES_DESC variable
Schedule data set (CPM), 80
E_START variable
Schedule data set (CPM), 61, 85, 95, 99, 101, 102
examples, *see* CPM examples
statement and option cross-reference tables (CPM), 301–303

F_FLOAT variable
Schedule data set (CPM), 61, 85, 101, 102
finish-before date
CPM procedure, 68
finish milestone
CPM procedure, 100, 101
example (CPM), 281
finish times
computation of, CPM procedure, 95
interpretation of, CPM procedure, 96
FINISH variable
Activity data set (CPM), 78
float
free, 61, 95, 102
total, 61, 95, 103
forced finish, *see* time constraints
forced start, *see* time constraints
formatting
details, CPM procedure, 141
free float, *see* float
FRI variable
Calendar data set (CPM), 105
functional summary
CPM procedure, 63
F_VAR variable
Schedule data set (CPM), 76, 192, 226

HEAD variable
Activity data set (CPM), 78
Holiday data set
CPM procedure, 57, 69, 79, 103, 106, 107, 163, 174
holidays example (CPM), 107
missing values, 139
treatment of holiday related variables, 79
variables, 137
holiday information, *see* Holiday data set
HOLIDAY variable

Holiday data set (CPM), 79, 106
holidays, *see* Holiday data set
defining, 79
durations, 79
finish times, 79
scheduling around, 61
start times, 79
HOLIDUR variable
Holiday data set (CPM), 79, 106
HOLIFIN variable
Holiday data set (CPM), 79, 106

ID variables
Activity data set (CPM), 80
independent resource scheduling, 86, 124
infeasibility diagnostics
CPM procedure, 86, 123, 214
input data sets, *see* Activity data set, *see* Calendar data set, *see* Holiday data set, *see* Resource data set, *see* Workday data set
CPM procedure, 57, 137

lag types, 92, 98
LAG variables, *see* nonstandard precedence relationships
Activity data set (CPM), 92
example (CPM), 183
late start schedule computation, *see* schedule computation
L_FINISH variable
Schedule data set (CPM), 61, 86, 95, 101, 102
LFINMILE variable
Schedule data set (CPM), 101
loop, *see* cycles
LS_ASC variable
Schedule data set (CPM), 80, 81
LS_DESC variable
Schedule data set (CPM), 80
L_START variable
Schedule data set (CPM), 61, 86, 95, 99, 101, 102

macro variable
ORCPM, 136
mandatory time constraints, *see* time constraints
maximum number of observations (CPM), 87
MAXNSEGMT variable
Activity data set (CPM), 86
memory requirements, *see* computer resource requirements
milestones
set finish (CPM), 71
MINSEGMTDUR variable
Activity data set (CPM), 87
missing values
CPM procedure, 139

MON variable
Calendar data set (CPM), 105
multiple alternates
CPM procedure, 87, 127
multiple calendars and holidays
common working calendar, 89
CPM procedure, 103–107, 111
example (CPM), 174
scheduling during common working times, 89
multiproject scheduling, 133–136
example (CPM), 248

negative requirements
specifying, CPM procedure, 121
negative resource requirements
example (CPM), 289, 292, 296
network diagrams, *see* NETDRAW procedure
Activity-on-Arc, 58
Activity-on-Node, 59
nonstandard precedence relationships
CPM procedure, 97, 98
example (CPM), 183
lag calendar, 92, 93, 184, 185
lag duration, 92
lag types, 92
lag variables, 92, 183
specification, CPM procedure, 92, 93

OBSTYPE variable
Resource data set (CPM), 88, 116
OBS_TYPE variable
Usage data set (CPM), 83
options classified by function, *see* functional summary
ORCPM macro variable, 136
output data sets, *see* Resource Schedule data set, *see* Schedule data set, *see* Usage data set
CPM procedure, 57
overview
CPM procedure, 57

PCTCOMP variable
Activity data set (CPM), 74, 112, 192
PCT_COMP variable
Schedule data set (CPM), 73
percent complete, 74, 112
PERIOD variable
Resource data set (CPM), 88, 96, 116
precedence diagramming method, *see* Activity-on-Node
precedence relationships
CPM procedure, 97, 98
scheduling subject to, 95, 96
problem size specification
CPM procedure, 141
number of activities (CPM), 69

number of adjacencies (CPM), 69
resource requirement array (CPM), 70
size of symbolic table (CPM), 69
utility data sets (CPM), 70, 141
progress updating
allow nonzero float, 74, 113
automatic updating of progress information, 73, 113, 192, 193
CPM procedure, 111–114
current time, 74, 112
estimate percent completion time, 73
example (CPM), 190
interrupt activities in progress, 74, 114
resource allocation during, 125
specifying information, 72–74
progress variables, 112
CPM procedure, 72, 112, 113
PROJ_DUR variable
Schedule data set (CPM), 136
project
definition, 58
finish date, 68, 95, 151
introductory example, 59
schedule comparison, 76
start date, 68, 95, 146, 151
project progress, *see* progress updating
PROJECT variable
Activity data set (CPM), 80, 133
Schedule data set (CPM, 136
PROJ_LEV variable
Schedule data set (CPM), 136

R_DELAY variable
Schedule data set (CPM), 84, 101, 203
remaining duration, 74, 112
REMDUR variable
Activity data set (CPM), 74, 192
replenishable resource, 117
RESID variable
Resource data set (CPM), 89, 116
resource allocation, *see* resource allocation control, 115, 116, 118–127, 130–133
activity splitting, 124, 125, 228
actual dates, 125
alternate resources, 118, 125, 126, 128, 233
analyzing infeasibilities, 86, 123, 214
auxiliary resources, 129
effect of the TIMENOW option, 125
example (CPM), 201
multiple alternates, 127
negative resource requirements, 121
progress updating, 125
resource availability profile, 119
resource calendars, 258
resource dictionary, 116
resource-driven durations, 124, 128
resource levels, 116
resource priority, 116, 118, 123
resource usage variables, 130
scheduling method, 121–123
scheduling rule, 90, 122, 201
secondary scheduling rule, 91, 122
specifying resource requirements, 120, 121
substitutable resources, 118, 125
supplementary levels, 116, 118
supplementary resources, 123
resource allocation control
activity-splitting options, 86, 87, 91
alternate resources, 83, 87, 89
checking levels needed, 86
cutoff date, 91
options related to, 83, 84, 86, 87, 89–91
scheduling rules, 90, 91
resource calendars
example (CPM), 258
resource constraints, *see* resource allocation
Resource data set
alternate resource specification, 126
CPM procedure, 57, 70, 116, 201
missing values, 139
multiple alternates, 127
observation type, 88
resource specification example (CPM), 119
variables, 116, 137
resource-driven durations, 92, 101, 114
and resource allocation (CPM), 124
CPM procedure, 128
example (CPM), 258, 296
resource infeasibilities, 86, 123, 214, 216, 222, 226
resource requirements
specifying, CPM procedure, 120, 121
Resource Schedule data set
CPM procedure, 57, 71
resource type
consumable, 117, 201
replenishable, 117
resource usage, *see* Usage data set
CPM procedure, 115, 131
cumulative usage, 84, 131
cumulative usage example, 210, 217
daily usage, 131
example of reports, 197
variables in Usage data set, 130
RESOURCE variables
Activity data set (CPM), 82, 120
example (CPM), 195
Resource data set (CPM), 82, 116
Resource Schedule data set (CPM), 133

R_RATE variable
Resource Schedule data set (CPM), 133
RSCHEDID variables
Activity data set (CPM), 90
Resource Schedule data set (CPM), 90

SAS data sets
CPM procedure, 57
SAS date, time, and datetime values
CPM procedure, 68, 96, 97, 107, 141
SAT variable
Calendar data set (CPM), 105
schedule computation
finish milestone, 100
multiproject, 135
nonstandard precedence constraints, 98
progress updating, 113
resource constraints, 121–123
standard precedence constraints, 95
time constraints, 98, 99
Schedule data set
CPM procedure, 57, 60, 70, 101–103, 126
multiproject, 136
options to control, 68, 72, 84–86, 88, 89, 91, 92
progress variables, 114
resources used, 126, 237
sort variables, 80, 81, 136, 249
variables, 61, 101–103, 114, 125, 126
schedule information, *see* Schedule data set
scheduling
around weekends and holidays, 61, 157, 161, 163, 169
rule for breaking ties, 91
rule for ordering activities, 90, 122, 123
serial parallel method, CPM procedure, 121, 122
secondary levels of resource
CPM procedure, 123
segments, *see* activity splitting
SEGMT_NO variable
Schedule data set (CPM), 103, 125, 231
serial-parallel scheduling method
CPM procedure, 121, 122
S_FINISH variable
Schedule data set (CPM), 101, 102, 125
SFINMILE variable
Schedule data set (CPM), 101
shift variables, *see* Workday data set
slack time, *see* float
sort variables
Schedule data set (CPM), 80, 81, 136, 249
split activities, *see* activity splitting
SS_ASC variable
Schedule data set (CPM), 80, 81
SS_DESC variable
Schedule data set (CPM), 80
S_START variable
Schedule data set (CPM), 101, 102, 125
start times
interpretation of, CPM procedure, 96
START variable
Activity data set (CPM), 78
STATUS variable
Resource Schedule data set (CPM), 133
Schedule data set (CPM), 101, 102, 114, 192
subprojects, *see* multiproject scheduling
individual critical paths for (CPM), 81
ordering activities within (CPM), 80, 81
substitutable resources, *see* alternate resources
SUCCESSOR variables
Activity data set (CPM), 92
SUN variable
Calendar data set (CPM), 105
supercritical activities
definition of, 95, 99
example of, 189
supplementary resources
CPM procedure, 123
example (CPM), 210
infinite levels, 214
SUPPL_R variable
Schedule data set (CPM), 84, 101, 203
S_VAR variable
Schedule data set (CPM), 76, 192, 226
syntax skeleton
CPM procedure, 63

table of syntax elements, *see* functional summary
TAIL variable
Activity data set (CPM), 93
target schedule, *see* baseline schedule
T_FLOAT variable
Schedule data set (CPM), 61, 91, 101, 103
THU variable
Calendar data set (CPM), 105
time-constrained scheduling
CPM procedure, 75, 96, 98, 99
example (CPM), 188
time constraints
activity align dates (CPM), 75, 188
activity align types (CPM), 75, 188
fix finish time (CPM), 69
Mandatory Finish, MF, 99, 189
Mandatory Start, MS, 99, 189
project finish date (CPM), 68, 69, 95
project start date (CPM), 68, 95
scheduling subject to, 98, 99
TIME variable

Usage data set (CPM), 83, 86–88, 90, 91, 130–132
TIMENOW, 74, *see* progress updating
allow activity splitting at, 74
total float, *see* float
TUE variable
Calendar data set (CPM), 105

units of duration
CPM procedure, 61
Usage data set
CPM procedure, 57, 71, 129, 131, 195, 197
description, 129, 130
options, 83–90
rate of usage, 83, 91
resource usage and availability profile, 131
total usage, 83, 91
variables, 129, 131, 132

variables
format of (CPM), 141
list of, CPM procedure, 137
treatment of missing values (CPM), 139

warning
suppress displaying (CPM), 72
WBS, 81, 82
WBS_CODE variable
Schedule data set (CPM), 82
WED variable
Calendar data set (CPM), 105
weekends and non-worked days
scheduling around, 61
WORK variable
Activity data set (CPM), 92, 114, 115
WORK variable
Resource Schedule data set (CPM), 133
workday
length of, 68, 96, 104, 105, 162
start of, 68, 96, 104, 105, 162
Workday data set
CPM procedure, 57, 72, 103, 104, 174
missing values, 139
shift variables, 104
variables, 137
work shift example (CPM), 104
workshift information, *see* Workday data set

Syntax Index

ACT statement, *see* ACTIVITY statement
ACTDELAY= option
    RESOURCE statement (CPM), 83, 123, 222
ACTIVITY statement
    CPM procedure, 59, 72, 144
ACTIVITYPRTY= option
    RESOURCE statement (CPM), 83
ACTPRTY keyword
    SCHEDRULE= option (CPM), 90, 122
ACTPRTY= option, *see* ACTIVITYPRTY= option
ACTUAL keyword
    COMPARE= option (CPM), 76
    SET= option (CPM), 77
    UPDATE= option (CPM), 77
ACTUAL statement
    CPM procedure, 72, 190
ADATE statement, *see* ALIGNDATE statement
ADDACT option
    PROC CPM statement, 67
ADDALLACT option, *see* ADDACT option
ADDCAL option
    RESOURCE statement (CPM), 83
ADDWBS option, *see* WBSCODE option
AF= option, *see* A_FINISH= option
A_FINISH= option
    ACTUAL statement (CPM), 73, 192
AGGREGATEPARENTRES option
    PROJECT statement (CPM), 80
AGGREGATEP_RES option, *see* AGGREGATEPARENTRES option
AGGREGPR option, *see* AGGREGATEPARENTRES option
ALAGCAL= option
    SUCCESSOR statement (CPM), 92, 185
ALIGN statement, *see* ALIGNTYPE statement
ALIGNDATE statement
    CPM procedure, 75, 188
ALIGNTYPE statement
    CPM procedure, 75, 188
ALL option, *see* ORDERALL option
    RESOURCE statement (CPM), 83, 130
ALTBEFORESUP option
    RESOURCE statement (CPM), 83, 237
ALTPRTY keyword
    OBSTYPE variable (CPM), 88, 89, 116, 118, 125
ALTRATE keyword
    OBSTYPE variable (CPM), 88, 89, 116, 118, 125
APPEND option
    RESOURCE statement (CPM), 83
APPENDINTXRATE option, *see* APPEND option
APPENDRATEXINT option, *see* APPEND option
APPENDUSAGE option, *see* APPEND option
AROUTCAL= option
    RESOURCE statement (CPM), 83, 130
AS= option, *see* A_START= option
A_START= option
    ACTUAL statement (CPM), 73, 192
ATYPE statement, *see* ALIGNTYPE statement
AUTOUPDT option
    ACTUAL statement (CPM), 73, 113, 193
AUXRES keyword
    OBSTYPE variable (CPM), 88, 89, 116, 118, 129
AVL option, *see* AVPROFILE option
AVP option, *see* AVPROFILE option
AVPROFILE option
    RESOURCE statement (CPM), 84, 130, 217
AWAITDELAY option
    RESOURCE statement (CPM), 84

BASELINE statement
    CPM procedure, 76, 190
BF= option, *see* B_FINISH= option
B_FINISH= option
    BASELINE statement (CPM), 76
BS= option, *see* B_START= option
B_START= option
    BASELINE statement (CPM), 76

CALEDATA= option
    PROC CPM statement, 67
CALENDAR keyword
    OBSTYPE variable (CPM), 88, 116, 118
CALENDAR= option, *see* CALEDATA= option, *see* CALEDATA= option
CALID statement
    CPM procedure, 77, 177
COLLAPSE option
    PROC CPM statement, 68, 183
COMPARE= option
    BASELINE statement (CPM), 76, 192
CPM procedure, 63
    ACTIVITY statement, 59, 72
    ACTUAL statement, 72, 190
    ALIGNDATE statement, 75, 188
    ALIGNTYPE statement, 75, 188
    BASELINE statement, 76, 190
    CALID statement, 77, 177, 181

DURATION statement, 58, 59, 78
HEADNODE statement, 58, 78
HOLIDAY statement, 79
ID statement, 80
PROC CPM statement, 67
PROJECT statement, 80
RESOURCE statement, 82, 195
SUCCESSOR statement, 59, 92
TAILNODE statement, 58, 93
CUMUSAGE option
RESOURCE statement (CPM), 84, 131, 210
CURRDATE= option, *see* TIMENOW= option

DATA= option
PROC CPM statement, 68, 146
DATE statement, *see* ALIGNDATE statement
DATE= option
PROC CPM statement, 68, 95, 146
DAY keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
DAYLENGTH= option
PROC CPM statement, 68, 96, 161
DAYSTART= option
PROC CPM statement, 68, 161
DELAY= option
RESOURCE statement (CPM), 84, 123, 210
DELAYANALYSIS option
RESOURCE statement (CPM), 84, 101, 201
DELAYLST keyword
SCHEDRULE= option (CPM), 90, 122
DESC option, *see* DESCENDING option
DESCENDING option
PROJECT statement (CPM), 80
DTDAY keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
DTHOUR keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
DTMINUTE keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
DTMONTH keyword
INTERVAL= option (CPM, 96
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
DTQTR keyword
INTERVAL= option (CPM, 96
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
DTSECOND keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
DTWEEK keyword
INTERVAL= option (CPM, 96
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
DTWRKDAY keyword
INTERVAL= option (CPM), 69, 96
ROUTINTERVAL= option (CPM), 90
DTYEAR keyword
INTERVAL= option (CPM, 96
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
DUR statement, *see* DURATION statement
DURATION statement
CPM procedure, 58, 59, 78, 144

EARLY keyword
COMPARE= option (CPM), 76
SET= option (CPM), 77
UPDATE= option (CPM), 77
ESO option, *see* ESORDER option
ESORDER option
PROJECT statement (CPM), 80
ESP option, *see* ESPROFILE option
ESPROFILE option
RESOURCE statement (CPM), 85, 130
ESS option, *see* ESPROFILE option
E_START option
RESOURCE statement (CPM), 85
ESTIMATEPCTC option
ACTUAL statement (CPM), 73
ESTPCTC option, *see* ESTIMATEPCTC option
ESTPCTCOMP option, *see* ESTIMATEPCTC option
ESTPROG option, *see* ESTIMATEPCTC option
EXCLUNSCHED option
RESOURCE statement (CPM), 85
EXPAND option, *see* ADDACT option

FBDATE= option
PROC CPM statement, 68, 95, 151
FEQ keyword
ALIGNTYPE variable (CPM), 75
FF keyword
LAG variable (CPM), 93
F_FLOAT option
RESOURCE statement (CPM), 85, 229
FGE keyword
ALIGNTYPE variable (CPM), 75
FILLMISSING option, *see* FILLUNSCHED option
FILLUNSCHED option
RESOURCE statement (CPM), 85
FINISH= option
DURATION statement (CPM), 78
FINISHBEFORE option
PROC CPM statement, 69

FIXASTART option
ACTUAL statement (CPM), 73
FIXFINISH option
PROC CPM statement, 69
FLE keyword
ALIGNTYPE variable (CPM), 75
FROM statement, *see* TAILNODE statement
FS keyword
LAG variable (CPM), 93

HDURATION= option, *see* HOLIDUR= option
HEAD statement, *see* HEADNODE statement
HEADNODE statement
CPM procedure, 58, 78, 148
HOLIDATA= option
PROC CPM statement, 69, 163
HOLIDAY statement
CPM procedure, 79, 163
HOLIDAY= option, *see* HOLIDATA= option
HOLIDAYS statement, *see* HOLIDAY statement
HOLIDUR= option
HOLIDAY statement (CPM), 79, 163
HOLIEND= option, *see* HOLIFIN= option
HOLIFIN= option
HOLIDAY statement (CPM), 79, 163
HOUR keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90

ID statement
CPM procedure, 80, 146
IGNOREPARENTRES option
PROJECT statement (CPM), 81
IGNOREPR option, *see* IGNOREPARENTRES option
IGNOREP_RES option, *see* IGNOREPARENTRES option
INCLUNSCHED option
RESOURCE statement (CPM), 85
INDEPALLOC option, *see* INDEPENDENTALLOC option
INDEPENDENTALLOC option
RESOURCE statement (CPM), 86, 124
INFEASDIAG option, *see* INFEASDIAGNOSTIC option
INFEASDIAGNOSTIC option
RESOURCE statement (CPM), 86, 123, 132, 214
INTERVAL= option
PROC CPM statement, 68, 69, 96, 97, 157
INTPER= option
PROC CPM statement, 69
INTUSAGE option, *see* TOTUSAGE option
INTXRATE option, *see* TOTUSAGE option

LAG= option
SUCCESSOR statement (CPM), 92, 93, 97, 183
LATE keyword
COMPARE= option (CPM), 76
SET= option (CPM), 77
UPDATE= option (CPM), 77
LFT keyword
SCHEDRULE= option (CPM), 90, 122
LSO option, *see* LSORDER option
LSORDER option
PROJECT statement (CPM), 81
LSP option, *see* LSPROFILE option
LSPROFILE option
RESOURCE statement (CPM), 86, 130
LSS option, *see* LSPROFILE option
LST keyword
SCHEDRULE= option (CPM), 90, 123
L_START option
RESOURCE statement (CPM), 86

MAXDATE= option
RESOURCE statement (CPM), 86, 130, 196
MAXNSEG= option, *see* MAXNSEGMT= option
MAXNSEGMT= option
RESOURCE statement (CPM), 86, 91, 124
MAXOBS= option
RESOURCE statement (CPM), 87
MF keyword
ALIGNTYPE variable (CPM), 75
MILESTONENORESOURCE
RESOURCE statement (CPM), 87
MILESTONERESOURCE
RESOURCE statement (CPM), 87
MINARATE keyword
OBSTYPE variable (CPM), 88, 116, 118, 127
MINDATE= option
RESOURCE statement (CPM), 87, 130
MINSEGD= option, *see* MINSEGMTDUR= option
MINSEGMTDUR= option
RESOURCE statement (CPM), 87, 91, 124, 231
MINUTE keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
MONTH keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
MS keyword
ALIGNTYPE variable (CPM), 75
MULTALT keyword
OBSTYPE variable (CPM), 88, 116, 118, 127
MULTALT option, *see* MULTIPLEALTERNATES option
MULTIPLEALTERNATES option
RESOURCE statement (CPM), 87, 126, 127

NACTS= option

PROC CPM statement, 69
NADJ= option
PROC CPM statement, 69
NLAGCAL= option
SUCCESSOR statement (CPM), 93
NNODES= option
PROC CPM statement, 69
NOAUTOUPDT option
ACTUAL statement (CPM), 73, 113, 192
NOE_START option
RESOURCE statement (CPM), 88
NOF_FLOAT option
RESOURCE statement (CPM), 88
NOL_START option
RESOURCE statement (CPM), 88
NORESOURCEVARS option
RESOURCE statement (CPM), 88
NORESVARS option, *see* NORESOURCEVARS option
NORESVARSOUT option, *see* NORESOURCEVARS option
NOT_FLOAT option
RESOURCE statement (CPM), 88
NOUTIL option
PROC CPM statement, 70
NRESREQ= option
PROC CPM statement, 70
NROUTCAL= option
RESOURCE statement (CPM), 88, 130

OBSTYPE= option
RESOURCE statement (CPM), 88, 201
ORDERALL option
PROJECT statement (CPM), 81
OUT= option
PROC CPM statement, 70, 148
OVERRIDEDUR option
DURATION statement (CPM), 78

PARENT statement, *see* PROJECT statement
PCOMP= option, *see* PCTCOMP= option
PCTCOMP= option
ACTUAL statement (CPM), 74, 192
PCTCOMPLETE= option, *see* PCTCOMP= option
PER= option, *see* PERIOD= option
PERIOD= option
RESOURCE statement (CPM), 88, 201
PROC CPM statement, *see* CPM procedure
statement options, 67
PROJECT statement
CPM procedure, 80

QTR keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90

RATEXINT option, *see* TOTUSAGE option
RCI option, *see* RESCALINTERSECT option
RCP option, *see* RCPROFILE option
RCPROFILE option
RESOURCE statement (CPM), 89, 130, 217
RCS option, *see* RCPROFILE option
RDUR= option, *see* REMDUR= option
RDURATION= option, *see* REMDUR= option
REMDUR= option
ACTUAL statement (CPM), 74, 192
RES statement, *see* RESOURCE statement
RESCALINT option, *see* RESCALINTERSECT option
RESCALINTERSECT option
RESOURCE statement (CPM), 89
RESID= option
RESOURCE statement (CPM), 89, 235
RESIN= option, *see* RESOURCEIN= option
RESLEVEL keyword
OBSTYPE variable (CPM), 88, 116, 117
RESLEVEL= option, *see* RESOURCEIN= option
RESOURCE keyword
COMPARE= option (CPM), 76
SET= option (CPM), 77
UPDATE= option (CPM), 77
RESOURCE statement
CPM procedure, 82, 195
RESOURCEIN= option
PROC CPM statement, 70, 201
RESOURCEOUT= option
PROC CPM statement, 71, 195
RESOURCESCHED= option
PROC CPM statement, 71
RESOURCEVARS option
RESOURCE statement (CPM), 89
RESOUT= option, *see* RESOURCEOUT= option
RESPRTY keyword
OBSTYPE variable (CPM), 88, 116, 118
SCHEDRULE= option (CPM), 90, 123
RESRCDUR keyword
OBSTYPE variable (CPM), 88, 116, 118
RESSCHED= option, *see* RESOURCESCHED= option
RESTYPE keyword
OBSTYPE variable (CPM), 88, 116, 117
RESUSAGE keyword
OBSTYPE variable (CPM), 88, 116, 117
RESUSAGE= option, *see* RESOURCEOUT= option
RESVARSOUT option, *see* RESOURCEVARS option
RIN= option, *see* RESOURCEIN= option
ROUT= option, *see* RESOURCEOUT= option
ROUTCONT option, *see* ROUTNOBREAK option
ROUTINTERVAL= option
RESOURCE statement (CPM), 90, 130

ROUTINTPER= option
RESOURCE statement (CPM), 90, 130, 131
ROUTNOBREAK option
RESOURCE statement (CPM), 90, 130, 210
RSCHDORD option, *see* RSCHEDORDER option
RSCHDWBS option, *see* RSCHEDWBS option
RSCHED= option, *see* RESOURCESCHED= option
RSCHEDID= option
RESOURCE statement (CPM), 90
RSCHEDORDER option
PROJECT statement (CPM), 81
RSCHEDULE= option, *see* RESOURCESCHED= option
RSCHEDWBS option
PROJECT statement (CPM), 81
RSID= option, *see* RSCHEDID= option
RSORDER option, *see* RSCHEDORDER option
RSWBS option, *see* RSCHEDWBS option
RULE2= option, *see* SCHEDRULE2= option
RULE= option, *see* SCHEDRULE= option

SCHEDRULE2= option
RESOURCE statement (CPM), 91, 122
SCHEDRULE= option
RESOURCE statement (CPM), 90, 122, 201
SECOND keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
SEPCRIT option
PROJECT statement (CPM), 81
SEQ keyword
ALIGNTYPE variable (CPM), 75
SET= option
BASELINE statement (CPM), 77, 190
SETFINISH= option
RESOURCE statement (CPM), 91
SETFINISHMILESTONE option
PROC CPM statement, 71
SF keyword
LAG variable (CPM), 93
SGE keyword
ALIGNTYPE variable (CPM), 75
SHORTDUR keyword
SCHEDRULE= option (CPM), 90, 123
SHOWFLOAT option
ACTUAL statement (CPM), 74, 113, 193
SLE keyword
ALIGNTYPE variable (CPM), 75
SLIPINF option, *see* DELAYANALYSIS option
SPLITFLAG option
RESOURCE statement (CPM), 91
SS keyword
LAG variable (CPM), 93
SSO option, *see* SSORDER option
SSORDER option
PROJECT statement (CPM), 81
START= option
DURATION statement (CPM), 78
STEP= option, *see* ROUTINTPER= option
STEPINT= option, *see* ROUTINTERVAL= option
STEPSIZE= option, *see* ROUTINTPER= option
STOPDATE= option
RESOURCE statement (CPM), 91, 245
SUCC statement, *see* SUCCESSOR statement
SUCCESSOR statement
CPM procedure, 59, 92, 144
SUPLEVEL keyword
OBSTYPE variable (CPM), 88, 116, 118
SUPPRESSOBSWARN option
PROC CPM statement, 72

TAIL statement, *see* TAILNODE statement
TAILNODE statement
CPM procedure, 58, 93, 148
T_FLOAT option
RESOURCE statement (CPM), 91, 229
TIMENOW= option
ACTUAL statement (CPM), 74, 192
TIMENOWSPLT option
ACTUAL statement (CPM), 74, 114
TO statement, *see* HEADNODE statement
TOTUSAGE option
RESOURCE statement (CPM), 91

UNSCHEDMISS option
RESOURCE statement (CPM), 92
UPDATE= option
BASELINE statement (CPM), 77
UPDTUNSCHED option
RESOURCE statement (CPM), 92
USEPROJDUR option
PROJECT statement (CPM), 82
USEPROJDURSPEC option, *see* USEPROJDUR option
USESPECDUR option, *see* USEPROJDUR option

WBS option, *see* WBSCODE option
WBSCODE option
PROJECT statement (CPM), 82
WEEK keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
WEEKDAY keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90
WORK= option
RESOURCE statement (CPM), 92
WORKDATA= option
PROC CPM statement, 72, 174

WORKDAY keyword
INTERVAL= option (CPM), 69, 96
ROUTINTERVAL= option (CPM), 90
WORKDAY= option, *see* WORKDATA= option

XFERVARS option
PROC CPM statement, 72, 188

YEAR keyword
INTERVAL= option (CPM), 69
ROUTINTERVAL= option (CPM), 90