Chapter 2

The CPM Procedure

Chapter Table of Contents

OVERVIEW 49

GETTING STARTED 50

SYNTAX 56
Functional Summary 56
PROC CPM Statement 60
ACTIVITY Statement 65
ACTUAL Statement 66
ALIGNDATE Statement 68
ALIGNTYPE Statement 69
BASELINE Statement 69
CALID Statement 71
DURATION Statement 71
HEADNODE Statement 72
HOLIDAY Statement 72
ID Statement 73
PROJECT Statement 73
RESOURCE Statement 76
SUCCESSOR Statement 85
TAILNODE Statement 87

DETAILS 87
Scheduling Subject to Precedence Constraints 88
Using the INTERVAL= Option 89
Nonstandard Precedence Relationships 91
Time-Constrained Scheduling 92
OUT= Schedule Data Set 93
Multiple Calendars 95
Baseline and Target Schedules 102
Progress Updating 103
Resource-Driven Durations and Resource Calendars 106
Resource Usage and Allocation 107
RESOURCEOUT= Usage Data Set 118
RESOURCESCHED= Resource Schedule Data Set 121
Multiproject Scheduling 121

Macro Variable _ORCPM_ 124
Input Data Sets and Related Variables 125
Missing Values in Input Data Sets 127
FORMAT Specification 128
Computer Resource Requirements 129

EXAMPLES 129
Example 2.1 Activity-on-Node Representation 131
Example 2.2 Activity-on-Arc Representation 135
Example 2.3 Meeting Project Deadlines 138
Example 2.4 Displaying the Schedule on a Calendar 140
Example 2.5 Precedence Gantt Chart 142
Example 2.6 Changing Duration Units 143
Example 2.7 Controlling the Project Calendar 146
Example 2.8 Scheduling around Holidays 149
Example 2.9 CALEDATA and WORKDATA data sets 154
Example 2.10 Multiple Calendars 158
Example 2.11 Nonstandard Relationships 168
Example 2.12 Activity Time Constraints 172
Example 2.13 Progress Update and Target Schedules 174
Example 2.14 Summarizing Resource Utilization 180
Example 2.15 Resource Allocation 184
Example 2.16 Using Supplementary Resources 192
Example 2.17 Use of the INFEASDIAGNOSTIC Option 197
Example 2.18 Variable Activity Delay 203
Example 2.19 Activity Splitting 209
Example 2.20 Alternate Resources 214
Example 2.21 PERT Assumptions and Calculations 220
Example 2.22 Scheduling Course - Teacher Combinations 223
Example 2.23 Resource Driven Durations and Resource Calendars 228
Example 2.24 Multiproject Scheduling 236
Statement and Option Cross-Reference Tables 245

REFERENCES 247

Chapter 2
The CPM Procedure

Overview

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 choose from a wide variety of options so that you can control the scheduling process. Thus, you may choose to allow project completion time to be delayed 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 6, “The PM Procedure,” for an interactive procedure that enables you to use a Graphical User Interface to enter and edit project information.

Getting Started

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.

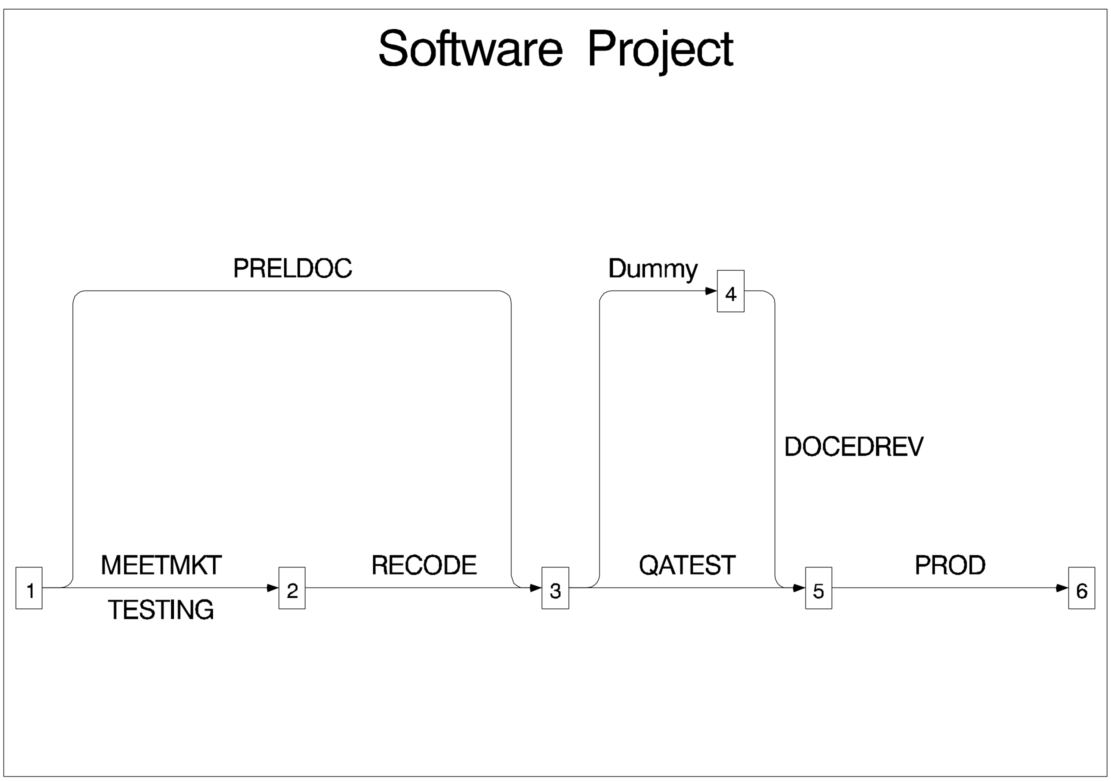


Figure 2.1. Activity-On-Arc Network

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 2.1 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 2.2 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* **;**

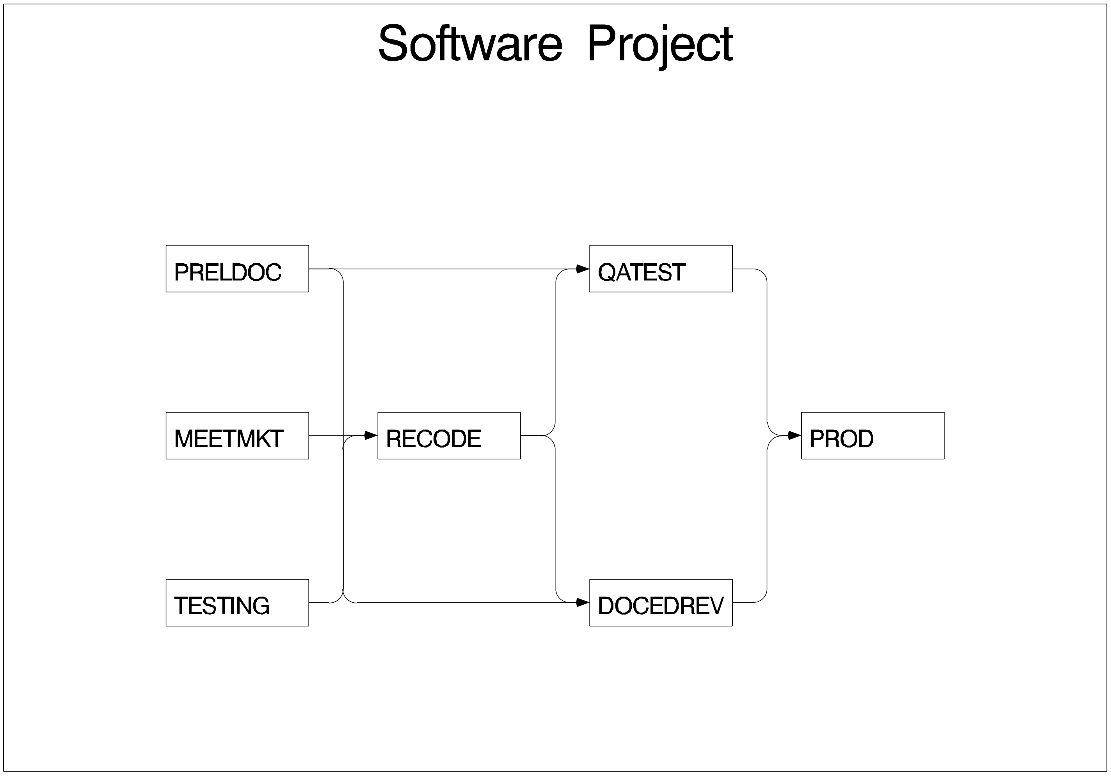


Figure 2.2. Activity-On-Node Network

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 via 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.

Among other things, the procedure determines

- 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 2.3.

The code for the entire program is as follows.

```
data software;
   input Descrpt  $char20.
         Duration 23-24
         Activity $ 27-34
         Succesr1 $ 37-44
         Succesr2 $ 47-54;
   datalines;
Initial Testing       20  TESTING   RECODE
Prel. Documentation   15  PRELDOC   DOCEDREV  QATEST
Meet Marketing        1   MEETMKT   RECODE
Recoding              5   RECODE    DOCEDREV  QATEST
QA Test Approve       10  QATEST    PROD
Doc. Edit and Revise  10  DOCEDREV  PROD
Production            1   PROD
;

proc cpm data=software
         out=intro1
         interval=day
         date='01mar92'd;
   id descrpt;
   activity activity;
   duration duration;
   successor succesr1 succesr2;
run;

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

```
                              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     01MAR92    20MAR92     01MAR92    20MAR92        0          0
 2     01MAR92    15MAR92     11MAR92    25MAR92       10         10
 3     01MAR92    01MAR92     20MAR92    20MAR92       19         19
 4     21MAR92    25MAR92     21MAR92    25MAR92        0          0
 5     26MAR92    04APR92     26MAR92    04APR92        0          0
 6     26MAR92    04APR92     26MAR92    04APR92        0          0
 7     05APR92    05APR92     05APR92    05APR92        0          0
```

Figure 2.3. Software Project Plan

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

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

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

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

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

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

T_FLOAT = L_START − E_START = L_FINISH − E_FINISH

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

In Figure 2.3 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 01MAR92 and 20MAR92.

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 choose different patterns of work during a day or a week (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 choosing 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 activites within a given project as well. 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 2.12 for more information on 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, allow 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 RESOURCESHCED= 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

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

The following tables outline the options available for the CPM procedure classified by function.

Table 2.1. Activity Splitting Specifications

Description	**Statement**	**Option**
split in-progress activities at TIMENOW	ACTUAL	TIMENOWSPLT
max. number of segments *variable*	RESOURCE	MAXNSEGMT=
min. segment duration *variable*	RESOURCE	MINSEGMTDUR=
allow splitting	RESOURCE	SPLITFLAG

Table 2.2. Baseline or Target Schedule Specifications

Description	**Statement**	**Option**
baseline finish date *variable*	BASELINE	B_FINISH=
baseline start date *variable*	BASELINE	B_START=
schedule to compare with baseline	BASELINE	COMPARE=
schedule to use as baseline	BASELINE	SET=
schedule to update baseline	BASELINE	UPDATE=

Table 2.3. Calendar Specifications

Description	**Statement**	**Option**
calendar *variable*	CALID	
holiday *variable*	HOLIDAY	
holiday duration *variable*	HOLIDAY	HOLIDUR=
holiday finish *variable*	HOLIDAY	HOLIFIN=

Table 2.4. Data Set Specifications

Description	**Statement**	**Option**
calendar input data set	CPM	CALEDATA=
activity input data set	CPM	DATA=
holiday input data set	CPM	HOLIDATA=
schedule output data set	CPM	OUT=
resource availability input data set	CPM	RESOURCEIN=
resource schedule output data set	CPM	RESOURCESCHED=
resource usage output data set	CPM	RESOURCEOUT=
workday input data set	CPM	WORKDATA=

Table 2.5. Duration Control Specifications

Description	**Statement**	**Option**
workday length	CPM	DAYLENGTH=
workday start	CPM	DAYSTART=
duration unit	CPM	INTERVAL=
duration multiplier	CPM	INTPER=
duration *variable*	DURATION	
finish *variable*	DURATION	FINISH=
override specified duration	DURATION	OVERRIDEDUR
start *variable*	DURATION	START=
work *variable*	RESOURCE	WORK=

Table 2.6. Lag Specifications

Description	**Statement**	**Option**
alphanumeric lag duration calendar	SUCCESSOR	ALAGCAL=
lag *variables*	SUCCESSOR	LAG=
numeric lag duration calendar	SUCCESSOR	NLAGCAL=

Table 2.7. Miscellaneous Options

Description	**Statement**	**Option**
suppress warning messages	CPM	SUPPRESSOBSWARN
fix L_FINISH for finish tasks to E_FINISH	CPM	FIXFINISH

Table 2.8. Network Specifications

Description	**Statement**	**Option**
AON format activity *variable*	ACTIVITY	
AOA format headnode *variable*	HEADNODE	
project *variable*	PROJECT	
AON format successor *variables*	SUCCESSOR	
AOA format tailnode *variable*	TAILNODE	

Table 2.9. Multiproject Specification

Description	Statement	Option
project *variable*	PROJECT	
aggregate parent resources	PROJECT	AGGREGATEPARENTRES
ignore parent resources	PROJECT	IGNOREPARENTRES
compute separate critical paths	PROJECT	SEPCRIT
use specified project duration	PROJECT	USEPROJDUR
compute WBS Code	PROJECT	WBSCODE

Table 2.10. OUT= Data Set Options

Description	Statement	Option
include percent complete variable	ACTUAL	ESTIMATEPCTC
add an observation for missing activities	CPM	ADDACT
single observation per activity	CPM	COLLAPSE
copy relevant variables to Schedule data set	CPM	XFERVARS
variables to be copied to Schedule data set	ID	
include descending sort variables	PROJECT	DESCENDING
include all sort order variables	PROJECT	ORDERALL
include early start sort order variable	PROJECT	ESORDER
include late start sort order variable	PROJECT	LSORDER
include resource start order variable	PROJECT	SSORDER
include WBS Code	PROJECT	WBSCODE
include information about resource delays	RESOURCE	DELAYANALYSIS
include early start schedule	RESOURCE	E_START
include free float	RESOURCE	F_FLOAT
set unscheduled S_START and S_FINISH	RESOURCE	FILLUNSCHED
include late start schedule	RESOURCE	L_START
exclude early start schedule	RESOURCE	NOE_START
exclude free float	RESOURCE	NOF_FLOAT
exclude late start schedule	RESOURCE	NOL_START
exclude resource variables	RESOURCE	NORESOURCEVARS
exclude total float	RESOURCE	NOT_FLOAT
include resource variables	RESOURCE	RESOURCEVARS
include total float	RESOURCE	T_FLOAT
set unscheduled S_START and S_FINISH to missing	RESOURCE	UNSCHEDMISS
update unscheduled S_START, S_FINISH	RESOURCE	UPDTUNSCHED

Table 2.11. Problem Size Options

Description	Statement	Option
number of precedence constraints	CPM	NADJ=
number of activities	CPM	NACTS=
number of distinct node or activity names	CPM	NNODES=
number of resource requirements	CPM	NRESREQ=

Table 2.12. Progress Updating Options

Description	Statement	Option
actual finish *variable*	ACTUAL	A_FINISH=
actual start *variable*	ACTUAL	A_START=
assume automatic completion	ACTUAL	AUTOUPDT
do not assume automatic completion	ACTUAL	NOAUTOUPDT
percentage complete *variable*	ACTUAL	PCTCOMP=
remaining duration *variable*	ACTUAL	REMDUR=
show float for all activities	ACTUAL	SHOWFLOAT
current date	ACTUAL	TIMENOW=

Table 2.13. Resource Variable Specifications

Description	Statement	Option
resource *variables*	RESOURCE	
observation type *variable*	RESOURCE	OBSTYPE=
resource availability date/time *variable*	RESOURCE	PERIOD=
alternate resource specification *variable*	RESOURCE	RESID=
work *variable*	RESOURCE	WORK=

Table 2.14. Resource Allocations Control Options

Description	Statement	Option
delay *variable*	RESOURCE	ACTDELAY=
activity priority *variable*	RESOURCE	ACTIVITYPRTY=
use alternate resources before supplementary levels	RESOURCE	ALTBEFORESUP
wait until L_START + DELAY	RESOURCE	AWAITDELAY
delay specification	RESOURCE	DELAY=
schedule even if insufficient resources	RESOURCE	INFEASDIAGNOSTIC
independent allocation	RESOURCE	INDEPENDENTALLOC
resource calendar intersect	RESOURCE	RESCALINTERSECT
scheduling priority rule	RESOURCE	SCHEDRULE=
secondary scheduling priority rule	RESOURCE	SCHEDRULE2=
stop date for resource constrained scheduling	RESOURCE	STOPDATE=

Table 2.15. RESOURCEOUT= Data Set Options

Description	Statement	Option
include all types of resource usage	RESOURCE	ALL
append observations for total usage	RESOURCE	APPEND
alphanumeric calendar for _TIME_	RESOURCE	AROUTCAL=
include availability profile for each resource	RESOURCE	AVPROFILE
cumulative usage for consumable resources	RESOURCE	CUMUSAGE
include early start profile for each resource	RESOURCE	ESPROFILE
exclude unscheduled activities in profile	RESOURCE	EXCLUNSCHED
include unscheduled activities in profile	RESOURCE	INCLUNSCHED
save observations for total usage	RESOURCE	TOTUSAGE
include late start profile for each resource	RESOURCE	LSPROFILE
maximum value of _TIME_	RESOURCE	MAXDATE=
maximum number of observations	RESOURCE	MAXOBS=
minimum value of _TIME_	RESOURCE	MINDATE=
numeric calendar for _TIME_	RESOURCE	NROUTCAL=
include resource constrained profile	RESOURCE	RCPROFILE
unit of difference between consecutive _TIME_ values	RESOURCE	ROUTINTERVAL=
difference between consecutive _TIME_ values	RESOURCE	ROUTINTPER=
use a continuous calendar for _TIME_	RESOURCE	ROUTNOBREAK

Table 2.16. RESOURCESCHED= Data Set Options

Description	Statement	Option
include WBS Code	PROJECT	RSCHEDWBS
include order variables	PROJECT	RSCHEDORDER
id *variables*	RESOURCE	RSCHEDID

Table 2.17. Time Constraint Specifications

Description	Statement	Option
alignment date *variable*	ALIGNDATE	
alignment type *variable*	ALIGNTYPE	
project start date	CPM	DATE=
project finish date	CPM	FBDATE=
finish before DATE= value	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 on the structure of this data set, see the "Multiple Calendars" section on page 95. 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. Note that 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 specifed 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 "Scheduling Subject to Precedence Constraints" section on page 88 for details on 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 "Multiple Calendars" section on page 95 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, WEEKDAYS, 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 "Using the INTERVAL= Option" section on page 89 for further information regarding this option.

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

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

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

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

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

OUT=*SAS-data-set*
specifies a name for the output data set that contains the schedule determined by PROC CPM. This data set (also referred to as the **Schedule** data set) contains all of the variables that were specified in the Activity data set to define the project. Every observation in the Activity data set has a corresponding observation in this output data set. If PROC CPM is used to determine a schedule that is not subject to any resource constraints, then this output data set contains the early and late start schedules; otherwise, it also contains the resource-constrained schedule. See the "OUT= Schedule Data Set" section on page 93 for information about the names of the new variables in the data set. If the OUT= option is omitted, the SAS system still 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 the resource rate affects the duration or not. 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 "RESOURCEIN= Input Data Set" section on page 107.

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 rest of this chapter. 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 for 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 "RESOURCEOUT= Usage Data Set" section on page 118.

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.

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 “Multiple Calendars” section on page 95 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 */ 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 "OUT= Schedule Data Set" section on page 93. See the "Progress Updating" section on page 103 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 time 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. Note that 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. Note that this value is 0 for activities that are 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.

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 "Progress Updating" section on page 103 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 "Progress Updating" section on page 103 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 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 "Progress Updating" section on page 103 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 issued 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 "Activity Splitting" section on page 115 for information regarding activity segments. Note that 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 "Time-Constrained Scheduling" section on page 92 for information on 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 2.18.

Table 2.18. 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 "Time-Constrained Scheduling" section on page 92 for information on 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.

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. Note that 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 issued 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 "Multiple Calendars" section on page 95 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. Note that 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 acitivity 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 the 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. Note that 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 "Multiproject Scheduling" section on page 121 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. Note that this order is not necessarily the same as the one that would be obtained by sorting all the activities in the Schedule data set by E_START.

IGNOREPARENTRES
IGNOREP_RES
IGNOREPR

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

LSORDER
LSO

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

ORDERALL
ALL

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

RSCHEDORDER
RSCHDORD
RSORDER

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

RSCHEDWBS
RSCHDWBS
RSWBS

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

SEPCRIT

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

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

SSORDER
SSO

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

USEPROJDUR
USEPROJDURSPEC
USESPECDUR

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

PROJ_DUR = E_FINISH - E_START

then the SEPCRIT option sets

L_FINISH <= E_START + PROJ_DUR

while the USEPROJDUR option sets

L_FINISH <= E_START + DUR

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

WBSCODE
WBS
ADDWBS

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

RESOURCE Statement

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

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

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

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

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

ACTIVITYPRTY=*variable*
ACTPRTY=*variable*
identifies the variable in the Activity data set that contains the priority of each activity. This option is required if resource-constrained scheduling is to be performed and the scheduling rule specified is ACTPRTY. If the value of the SCHEDRULE= option is specified as the keyword ACTPRTY, then all activities waiting for resources are ordered by increasing values of the ACTPRTY= variable. Missing values of the activity priority variable are treated as +INFINITY. See the "Scheduling Method" section on page 111 for a description of the various scheduling rules used during resource constrained scheduling.

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 “RESOURCEOUT= Usage Data Set” section on page 118 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 1JUN92 and L_START is 5JUN92 and *delay* is specified as 2, PROC CPM first tries to schedule the activity to start on 1JUN92. 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 7JUN92 (because *delay*=2 and L_START=5JUN92).

If the activity cannot be scheduled even on 7JUN92, 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 "Scheduling Method" section on page 111 for a definition of this waiting list of activities.) Note that 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. Note that 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 allows the exclusion of activities that are still unscheduled. Note that the unscheduled activites 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. Note that if resource allocation is not performed, this variable is always included in the output data set.

INCLUNSCHED

allows 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

allows each resource to be scheduled independently for each activity during resource-constrained scheduling. Consider the basic resource scheduling algorithm described in the “Scheduling Method” section on page 111. 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.

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.

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 behaviour. To include the F_FLOAT variable in addition to the resource-constrained schedule, use the F_FLOAT option. Note that 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. Note that 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, RESPRTY, SUPLEVEL, ALTRATE, ALTPRTY, RESRCDUR, and CALENDAR. 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
RCPS

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.

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* is being specified in that observation. Such observations 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. Note that the specification of the RESID= option triggers the use of alternate resources. See the "Specifying Alternate Resources" section on page 116 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. Note that *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. Note that, 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 "Scheduling Rules" section on page 112 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. Note that 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.

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 succesive 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 "Resource-Driven Durations and Resource Calendars" section on page 106.

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*

specifes 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 2.19 contains some examples of lag specifications.

Table 2.19. 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*
specifes 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

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 “Time-Constrained Scheduling” section on page 92, the statements needed to place time constraints on the activities are introduced. The “OUT= Schedule Data Set” section on page 93 describes the format of the schedule output data set (the Schedule data set). The “Multiple Calendars” section on page 95 deals with calendar specifications for the different activities.

The “Baseline and Target Schedules” section on page 102 describes how you can save specific schedules as baseline or target schedules. The “Progress Updating” section on page 103 describes how to incorporate the actual start and finish times for a project that is already in progress. The “Resource-Driven Durations and Resource Calendars” section on page 106 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 “Resource Usage and Allocation” section on page 107 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 are discussed under separate subsections. The “RESOURCEOUT= Usage Data Set” section on page 118 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 "RESOURCESCHED= Resource Schedule Data Set" section on page 121, contains the individual resource schedules for each activity.

The "Multiproject Scheduling" section on page 121 describes how you can use PROC CPM when there are multiple projects that have been combined together in a multi-project structure.

PROC CPM also defines a macro variable that is described in the "Macro Variable _ORCPM_" section on page 124. Table 2.24 in the "Input Data Sets and Related Variables" section on page 125 lists all the variables used by the CPM procedure and the data sets that contain them. Table 2.25 in the "Missing Values in Input Data Sets" section on page 127 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 "FORMAT Specification" section on page 128 underlines the importance of associating the correct FORMAT specification with all the date-type variables, and the "Computer Resource Requirements" section on page 129 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.

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 2.20 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.

Table 2.20. 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

For the first five specifications of the INTERVAL= option in the last part of Table 2.20 (DTDAY , . . . , DTHOUR), the day starts at *daystart* and is *daylength* hours long.

Note that 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.

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 allow 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 “SUCCESSOR Statement” section on page 85 for information on the specification. Example 2.11 , “Non-Standard Relationships,” in the “Examples” section illustrates a nonstandard precedence relationship.

This section briefly discusses how the computation of the early and late start schedules, described in the “Scheduling Subject to Precedence Constraints” section on page 88, 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 “Scheduling Subject to Precedence Constraints” section on page 88 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 2.21 lists the predecessor’s value that is used to determine the successor’s early start time.

Table 2.21. 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

Note that 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 2.22 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 2.22. 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 2.22, 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 2.21 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 L_START times of all its successors.

It is important to remember that the precedence constraints of the network are always respected. 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.

Note that 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 that make the target completion date (or constraints on the finish times of some successor activities) 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 2.12, "Activity Time Constraints," for a demonstration of this situation.

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 "Resource-Driven Durations and Resource Calendars" section on page 106.

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 "Progress Updating" section on page 103 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.

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.

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 "Activity Splitting" section on page 115.

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 allow for 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 nonworking 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. 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 “WORKDATA Data Set” section on page 97. 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. Note that 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 the HOLIFIN variables, only the HOLIFIN variable is used so that the holiday is assumed to start and end as specified by the HOLIDAY and HOLIFIN variables, respectively. 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 23DEC87 to 25DEC87 (both days inclusive). Note that 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_

23DEC87         .         25DEC87       .
01JAN88         1            .          1
18JAN88         .            .          2
28JAN88         3            .          2
28JAN88         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 2.23.

Note that, 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 28, 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 2.23. Holiday Definitions

Calendar	Holiday Start	Holiday End
0	23DEC87:09:00	25DEC87:16:59:59
1	23DEC87:09:00	25DEC87:07:59:59
	01JAN88:00:00	01JAN88:07:59:59
2	23DEC87:09:00	25DEC87:16:59:59
	18JAN88:09:00	18JAN88:16:59:59
	28JAN88:09:00	01FEB88:16:59:59
3	23DEC87:09:00	25DEC87:16:59:59
	28JAN88:09:00	30JAN88:16:59:59

You can use the GANTT procedure to visualize the breaks and holidays for the different calendar. Figure 2.4 shows all the breaks and holidays for the period between Christmas and New Year. Holidays and breaks are denoted by *. Likewise, Figure 2.5 shows the vacation periods in January for calendars '2' and '3'.

```
                        Christmas and New Year Holidays

           DEC         DEC         DEC         DEC         DEC         DEC
           22          22          23          23          24          24
  _cal_    00:00       12:00       00:00       12:00       00:00       12:00
          -+-----------+-----------+-----------+-----------+-----------+-
      0   |*********--------********************************************|
      1   |*******----*----*********************************************|
      2   |*********--------********************************************|
      3   |*********--------********************************************|
          -+-----------+-----------+-----------+-----------+-----------+-
                         Christmas and New Year Holidays

 DEC         DEC         DEC         DEC         DEC         DEC         DEC
 24          25          25          26          26          27          27
 12:00       00:00       12:00       00:00       12:00       00:00       12:00
-+-----------+-----------+-----------+-----------+-----------+-----------+-
|*********************************************--------****************----|
|*******************************************----**************************|
|*************************************************************************|
|*********************************************--------****************----|
-+-----------+-----------+-----------+-----------+-----------+-----------+-
                         Christmas and New Year Holidays

 DEC         DEC         DEC         DEC         DEC         DEC         DEC
 27          28          28          29          29          30          30
 12:00       00:00       12:00       00:00       12:00       00:00       12:00
-+-----------+-----------+-----------+-----------+-----------+-----------+-
|-----****************--------****************--------****************----|
|*******************----*----***************----*----*****************----|
|*********************--------****************--------****************----|
|-----****************--------****************--------****************----|
-+-----------+-----------+-----------+-----------+-----------+-----------+-
                         Christmas and New Year Holidays

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

Figure 2.4. Christmas and New Year Holidays for Multiple Calendars

```
                            Vacation Times for Calendars 2 and 3

            JAN         JAN         JAN         JAN         JAN         JAN         JAN
            18          18          19          19          20          20          21
 _cal_      00:00       12:00       00:00       12:00       00:00       12:00       00:00
           -+-----------+-----------+-----------+-----------+-----------+-----------+-
     2     |*********************************--------****************--------********|
     3     |*********--------****************--------****************--------********|
           -+-----------+-----------+-----------+-----------+-----------+-----------+-
                             Vacation Times for Calendars 2 and 3

          JAN         JAN         JAN         JAN         JAN         JAN         JAN
          21          21          22          22          23          23          24
          00:00       12:00       00:00       12:00       00:00       12:00       00:00
         -+-----------+-----------+-----------+-----------+-----------+-----------+-
         |*********--------****************--------*********************************|
         |*********--------****************--------****************--------********|
         -+-----------+-----------+-----------+-----------+-----------+-----------+-
                             Vacation Times for Calendars 2 and 3

          JAN         JAN         JAN         JAN         JAN         JAN         JAN
          24          24          25          25          26          26          27
          00:00       12:00       00:00       12:00       00:00       12:00       00:00
         -+-----------+-----------+-----------+-----------+-----------+-----------+-
         |*********************************--------****************--------********|
         |*********--------****************--------****************--------********|
         -+-----------+-----------+-----------+-----------+-----------+-----------+-
                             Vacation Times for Calendars 2 and 3

          JAN         JAN         JAN         JAN         JAN         JAN         JAN
          27          27          28          28          29          29          30
          00:00       12:00       00:00       12:00       00:00       12:00       00:00
         -+-----------+-----------+-----------+-----------+-----------+-----------+-
         |*********--------********************************************************|
         |*********--------********************************************************|
         -+-----------+-----------+-----------+-----------+-----------+-----------+-
                             Vacation Times for Calendars 2 and 3

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

Figure 2.5. Vacation Time for Calendars 2 and 3

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 “BASELINE Statement” section on page 69 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 options that are discussed in this section are:

- 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 issued 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 nonworking 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 PCT, 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 already started activities. 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 are 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 issued 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 "RESOURCEIN= Input Data Set" section on page 107) 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 "Specifying Resource Requirements" section on page 110, and the description of the format of the Resource data set is given in the "RESOURCEIN= Input Data Set" section, which follows. The "Scheduling Method" section on page 111 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*.

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, RESPRTY, SUPLEVEL, ALTPRTY, ALTRATE, RESRCDUR, and CALENDAR. 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. Note that, 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.

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 "Scheduling Rules" section on page 112). 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 "Secondary Levels of Resources" section on page 114 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 or ALTPRTY, 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. See the "Specifying Alternate Resources" section on page 116 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. Note that this use of the calendar requires that the calendar variable in the Activity and other data sets be numeric.

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 example illustrates the details about the Resource data set. Consider the following Resource data:

OBS	OBSTYPE	DATE	WORKERS	BRICKS
1	RESTYPE	.	1	2
2	RESPRTY	.	10	10
3	SUPLEVEL	.	1	.
4	RESLEVEL	1JUL92	.	1000
5	RESLEVEL	5JUL92	4	.
6	RESLEVEL	9JUL92	.	1500

There are two resources in these data, WORKERS and BRICKS. 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 and BRICKS is a consumable resource. The second observation indicates that both resources have equal priority. In the third 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 resource BRICKS.

The next three observations indicate the resource availability profile. The resource WORKERS is unavailable until July 5, 1992, 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, 1992, at level '1000'. On July 9, an additional 500 bricks are made available to increase the total availability to 1500. Note that 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. Note that 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 the 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.

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'.

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 "Scheduling Rules" section on page 112 for details on 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.

Note that 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 2.16, "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 nonworking 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 "Activity Splitting" section on page 115.

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 the "Scheduling Method" section on page 111, 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 that they use. 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 allow the project completion to be delayed 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 choose 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 2.17, “Use of the INFEASDIAGNOSTIC Option,” for an illustration.

Note that 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 allows 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. Note that the activity is considered to be finished (and its successors can start) only after all the resources for that activity have been scheduled. Note also that this option is valid even if all activities have fixed durations and calendars are not associated with resources.

Activity Splitting

As mentioned in the "Scheduling Method" section on page 111, 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 obervations 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 issued 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 on supplementary resources, see the “Secondary Levels of Resources” section on page 114.

If activity splitting is allowed (either 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 “Scheduling Method” section on page 111.

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 via 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, AL-

TRATE 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 via 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		15FEB91	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 allows 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 example above). 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). Let OBSTYPE = 'ALTPRTY' for the observation that specifies a prioritization for the resources. Note that 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. Note that 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.

See Example 2.20 for an illustration of the use of alternate resources.

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 "RESOURCE Statement" section on page 76) 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 chooses 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 according as 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 "Specifying Resource Requirements" section on page 110), activities 'A' and 'B' have S_START equal to 1JUL92 and 5JUL92, 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
1JUL92	0	0	100	1000
2JUL92	0	0	100	900
3JUL92	0	0	100	800
4JUL92	0	0	100	700
5JUL92	2	2	100	600
6JUL92	2	2	0	500
7JUL92	2	2	0	500
8JUL92	2	2	0	500
9JUL92	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, 1992), all 1000 bricks are still available. Note that 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
1JUL92	0	0	0	1000
3JUL92	0	0	200	800
5JUL92	2	2	400	600
7JUL92	2	2	500	500
9JUL92	0	4	500	1000

Note that 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 1JUL92 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 11JUL92, it would indicate that 100 bricks from the supplementary reservoir were used through the end of July 10, 1992. See Example 2.16, “Using Supplementary Resources,” and Example 2.17, “Use of the INFEASDIAGNOSTIC Option.”

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 succesive observations*. The observations (five in each set) would be as follows:

TIME	OBS_TYPE	RWORKERS	RBRICKS
01JUL92	RES_RATE	0	100
03JUL92	RES_RATE	0	100
05JUL92	RES_RATE	2	100
07JUL92	RES_RATE	2	0
09JUL92	RES_RATE	0	0
01JUL92	RES_USED	0	200
03JUL92	RES_USED	0	200
05JUL92	RES_USED	4	100
07JUL92	RES_USED	4	0
09JUL92	RES_USED	0	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. Note that 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. Note that 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."

Once you have specified resource requirements for supertasks, you can control how this information is used by the scheduling algorithm in a couple of different ways. You can use the AGGREGATEPARENTRES 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 choose to ignore any resource requirements specified for supertasks by specifying the IGNOREPARENTRES option. Note the difference between the AGGREGPARENTRES 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. Note that 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. Note that 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 minumum 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. Note that 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; note that 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= REASON=, where STATUS= is either SUCCESSFUL or ERROR_EXIT and REASON= (if PROC CPM terminated unsuccessfully) can be one of the following:

- CYCLE
- RES_INFEASIBLE
- BADDATA_ERROR
- MEMORY_ERROR
- IO_ERROR
- SEMANTIC_ERROR
- SYNTAX_ERROR
- CPM_BUG
- UNKNOWN_ERROR

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 2.24 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 2.24. 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

Table 2.24. (continued)

Data Set	Statement	Variable Name	Interpretation
		PCTCOMP	Percentage of work completed
	ALIGNDATE	ALIGNDATE	Time constraint on activity
	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

Table 2.24. (continued)

Data Set	Statement	Variable Name	Interpretation
	TAILNODE	TAILNODE	Tail of arrow (arc) in AOA format
HOLIDATA	CALID	CALID	Calendar to which holiday applies
	HOLIDAY	HOLIDAY HOLIDUR HOLIFIN	Start of holiday Duration of holiday End of holiday
RESOURCEIN	RESOURCE	OBSTYPE	Type of observation; valid values: RESLEVEL, RESTYPE, SUPLEVEL, RESPRTY, ALTRATE, ALTPRTY
		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 2.25. 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
	ALIGNTYPE	SGE: if ALIGNDATE is not missing
	A_FINISH	see “Progress Updating” for details
	A_START	see “Progress Updating” for details

Table 2.25. (continued)

Data Set	Variable	Value Used / Assumption Made / Action Taken
	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 "Progress Updating" for details
	PROJECT	activity is at highest level
	REMDUR	see "Progress Updating" 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 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. 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 “Syntax” section on page 56 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

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 2.28 and Table 2.29, 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 2.22 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 two examples use different projects to illustrate resource-driven durations and multiproject scheduling.

There are 14 activities in the widget manufacturing project. Example 2.1 and Example 2.2 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 2.26 lists the detailed names (and corresponding abbreviations) of all the activities in the project and the department that is responsible for each one.

Table 2.26. Widget Manufacture: Activity List

Task	Department	Activity Description
Approve Plan	Planning	Finalize and Approve Plan
Drawings	Engineering	Prepare Drawings
Anal. 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 2.27. Widget Manufacture: Precedence Information

Task	Dur	Successor	Successor	Successor
Approve Plan	10	Drawings	Anal. Market	Write Specs
Drawings	20	Prototype		
Anal. 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			

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’ and ‘Anal. Market’ and ‘Write Specs’ can start. Table 2.27 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.

The relationship among the tasks can be represented by the network in Figure 2.6. The diagram was produced by the NETDRAW procedure. The code used is the same as in Example 5.11 in Chapter 5, “The NETDRAW Procedure” (except for the colors, which may be different).

Example 2.1. Activity-on-Node Representation

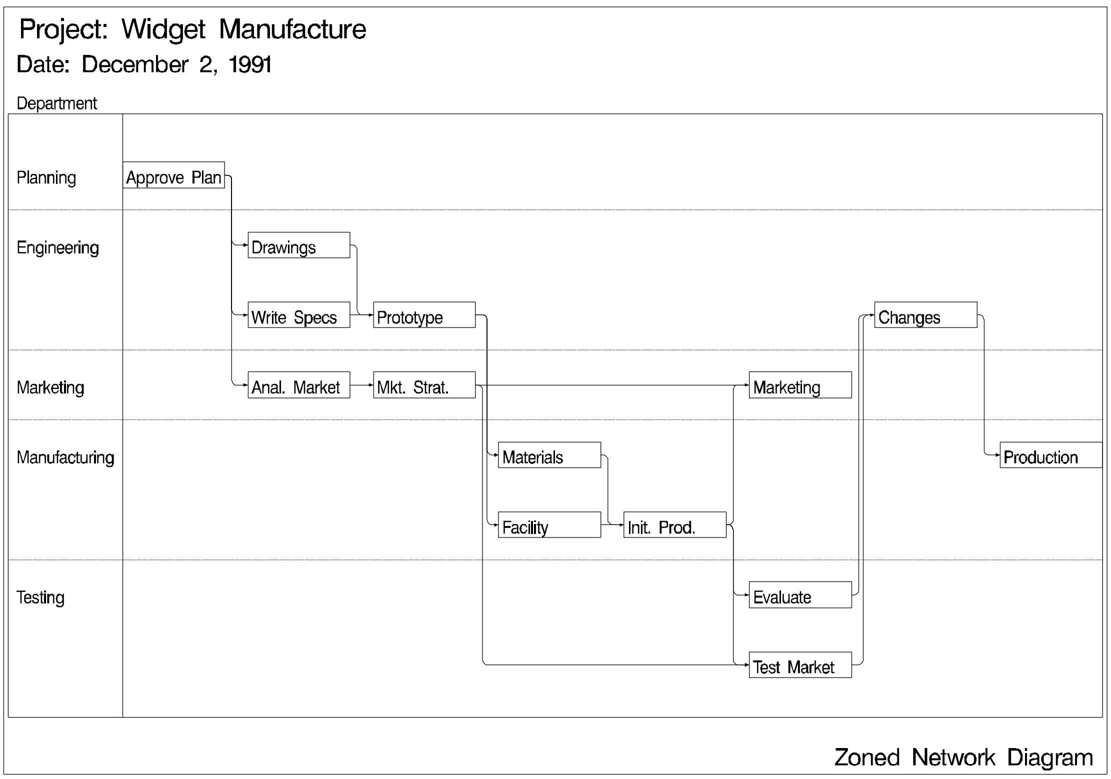


Figure 2.6. Network Showing Task Relationships in Activity-on-Node Format

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 2.1.1. 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;
   input task $ 1-12 days succ1 $ 19-30 succ2 $ 33-44 succ3 $ 47-58;
   datalines;
Approve Plan   5  Drawings      Anal. Market  Write Specs
Drawings      10  Prototype
Anal. 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 2.1.1. Critical Path

```
              Widget Manufacture: Activity-On-Node Format
                             Critical Path

                                                            E     L
                                                         E  _  L  _  T  F
                                                         _  F  _  F  _  _
                 s           s            s              S  I  S  I  F  F
  t              u           u            u          d   T  N  T  N  L  L
O a              c           c            c          a   A  I  A  I  O  O
b s              c           c            c          y   R  S  R  S  A  A
s k              1           2            3          s   T  H  T  H  T  T

 1 Approve Plan Drawings    Anal. Market Write Specs  5  0  5  0  5  0  0
 2 Drawings     Prototype                            10  5 15  5 15  0  0
 3 Anal. 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 2, 1991, 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 output data set. Output 2.1.2 displays the resulting output data set.

```
data details;
   input task $ 1-12 dept $ 15-27 descrpt $ 30-59;
   label dept =  "Department"
         descrpt = "Activity Description";
   datalines;
Approve Plan  Planning       Finalize and Approve Plan
Drawings      Engineering    Prepare Drawings
Anal. 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 2, 1991  */
proc cpm data=widgetn date='2dec91'd;
   activity task;
   successor succ1 succ2 succ3;
   duration days;
   id dept descrpt;
   run;

proc sort;
   by e_start;
   run;

options ls=90;
```

```
title2 'Project Schedule';
proc print;
   id descrpt;
   var dept e_: l_: t_float f_float;
   run;
```

Output 2.1.2. Critical Path: Activity-On-Node Format

```
                         Widget Manufacture: Activity-On-Node Format
                                      Project Schedule

                                                           E                 L
d                                                 E        _        L        _     T   F
e                                                 _        F        _        F     _   _
s                                                 S        I        S        I     F   F
c                               d                 T        N        T        N     L   L
r                               e                 A        I        A        I     O   O
p                               p                 R        S        R        S     A   A
t                               t                 T        H        T        H     T   T

Finalize and Approve Plan       Planning       02DEC91  06DEC91  02DEC91  06DEC91   0   0
Prepare Drawings                Engineering    07DEC91  16DEC91  07DEC91  16DEC91   0   0
Analyze Potential Markets       Marketing      07DEC91  11DEC91  06JAN92  10JAN92  30   0
Write Specifications            Engineering    07DEC91  11DEC91  12DEC91  16DEC91   5   5
Develop Marketing Concept       Marketing      12DEC91  21DEC91  11JAN92  20JAN92  30  30
Build Prototype                 Engineering    17DEC91  31DEC91  17DEC91  31DEC91   0   0
Procure Raw Materials           Manufacturing  01JAN92  10JAN92  01JAN92  10JAN92   0   0
Prepare Manufacturing Facility  Manufacturing  01JAN92  10JAN92  01JAN92  10JAN92   0   0
Initial Production Run          Manufacturing  11JAN92  20JAN92  11JAN92  20JAN92   0   0
Evaluate Product In-House       Testing        21JAN92  30JAN92  26JAN92  04FEB92   5   5
Test Product in Sample Market   Testing        21JAN92  04FEB92  21JAN92  04FEB92   0   0
Begin Full Scale Marketing      Marketing      21JAN92  21JAN92  10FEB92  10FEB92  20  20
Engineering Changes             Engineering    05FEB92  09FEB92  05FEB92  09FEB92   0   0
Begin Full Scale Production     Manufacturing  10FEB92  10FEB92  10FEB92  10FEB92   0   0
```

Example 2.2. Activity-on-Arc Representation

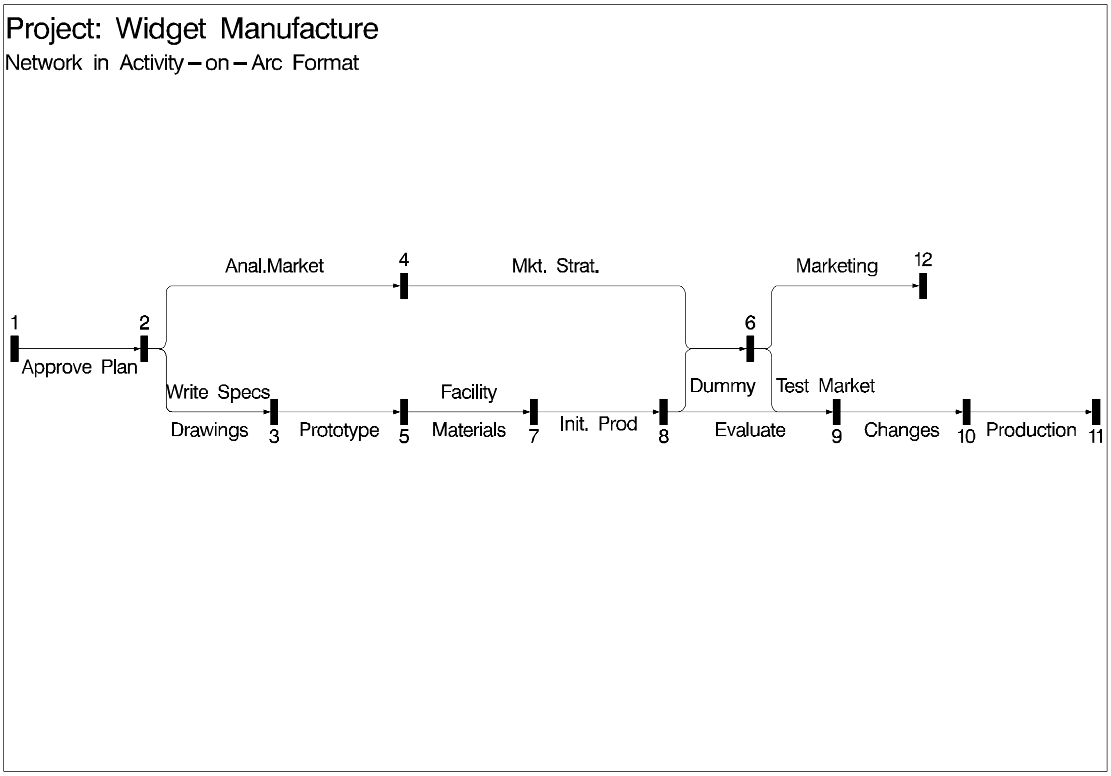


Figure 2.7. Network Showing Task Relationships in Activity-on-Arc Format

The problem discussed in Example 2.1 can also be described in an AOA format. The network is illustrated in Figure 2.7. Note that the network has an arc labeled 'Dummy', which is required to capture accurately 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. Note that 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 2.1, 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 2.2.1 displays the project schedule.

```
/* Activity-on-Arc representation of the project */
  data widgaoa;
     input task $ 1-12 days tail head;
     datalines;
  Approve Plan   5   1   2
  Drawings      10   2   3
  Anal. 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;
     input task $ 1-12 dept $ 15-27 descrpt $ 30-59;
     label dept = "Department"
           descrpt = "Activity Description";
     datalines;
```

```
Approve Plan  Planning       Finalize and Approve Plan
Drawings      Engineering    Prepare Drawings
Anal. 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='2dec91'd out=save;
   tailnode tail;
   headnode head;
   duration days;
   activity task;
   id dept descrpt;
   run;

proc sort;
   by e_start;
   run;

options ls=90;

title 'Widget Manufacture: Activity-On-Arc Format';
title2 'Project Schedule';
proc print;
   id descrpt;
   var dept e_: l_: t_float f_float;
   run;
```

Output 2.2.1. Critical Path: Activity-on-Arc Format

Widget Manufacture: Activity-On-Arc Format
Project Schedule

descript	dept	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
Finalize and Approve Plan	Planning	02DEC91	06DEC91	02DEC91	06DEC91	0	0
Prepare Drawings	Engineering	07DEC91	16DEC91	07DEC91	16DEC91	0	0
Analyze Potential Markets	Marketing	07DEC91	11DEC91	06JAN92	10JAN92	30	0
Write Specifications	Engineering	07DEC91	11DEC91	12DEC91	16DEC91	5	5
Develop Marketing Concept	Marketing	12DEC91	21DEC91	11JAN92	20JAN92	30	30
Build Prototype	Engineering	17DEC91	31DEC91	17DEC91	31DEC91	0	0
Procure Raw Materials	Manufacturing	01JAN92	10JAN92	01JAN92	10JAN92	0	0
Prepare Manufacturing Facility	Manufacturing	01JAN92	10JAN92	01JAN92	10JAN92	0	0
Initial Production Run	Manufacturing	11JAN92	20JAN92	11JAN92	20JAN92	0	0
Evaluate Product In-House	Testing	21JAN92	30JAN92	26JAN92	04FEB92	5	5
Mail Product to Sample Market	Testing	21JAN92	04FEB92	21JAN92	04FEB92	0	0
Begin Full Scale Marketing	Marketing	21JAN92	21JAN92	10FEB92	10FEB92	20	20
Production Milestone		21JAN92	21JAN92	21JAN92	21JAN92	0	0
Engineering Changes	Engineering	05FEB92	09FEB92	05FEB92	09FEB92	0	0
Begin Full Scale Production	Manufacturing	10FEB92	10FEB92	10FEB92	10FEB92	0	0

Example 2.3. 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 2.3.1 shows the resulting schedule. Note that the entire schedule is shifted in time (as compared to the schedule in Output 2.2.1) so that the end of the project is on March 1, 1992. 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 2.3.2 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='1mar92'd interval=day;
   tailnode tail;
   headnode head;
   duration days;
   id task;
   run;

proc sort;
   by e_start;
run;
```

```
options ps=60 ls=78;

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='1mar92'd
         date='2dec91'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 2.3.1. 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	22DEC91	26DEC91	22DEC91	26DEC91	0	0
Drawings	27DEC91	05JAN92	27DEC91	05JAN92	0	0
Anal. Market	27DEC91	31DEC91	26JAN92	30JAN92	30	0
Write Specs	27DEC91	31DEC91	01JAN92	05JAN92	5	5
Mkt. Strat.	01JAN92	10JAN92	31JAN92	09FEB92	30	30
Prototype	06JAN92	20JAN92	06JAN92	20JAN92	0	0
Materials	21JAN92	30JAN92	21JAN92	30JAN92	0	0
Facility	21JAN92	30JAN92	21JAN92	30JAN92	0	0
Init. Prod.	31JAN92	09FEB92	31JAN92	09FEB92	0	0
Evaluate	10FEB92	19FEB92	15FEB92	24FEB92	5	5
Test Market	10FEB92	24FEB92	10FEB92	24FEB92	0	0
Marketing	10FEB92	10FEB92	01MAR92	01MAR92	20	20
Dummy	10FEB92	10FEB92	10FEB92	10FEB92	0	0
Changes	25FEB92	29FEB92	25FEB92	29FEB92	0	0
Production	01MAR92	01MAR92	01MAR92	01MAR92	0	0

Output 2.3.2. 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   02DEC91   06DEC91    22DEC91   26DEC91       20         0
Drawings       07DEC91   16DEC91    27DEC91   05JAN92       20         0
Anal. Market   07DEC91   11DEC91    26JAN92   30JAN92       50         0
Write Specs    07DEC91   11DEC91    01JAN92   05JAN92       25         5
Mkt. Strat.    12DEC91   21DEC91    31JAN92   09FEB92       50        30
Prototype      17DEC91   31DEC91    06JAN92   20JAN92       20         0
Materials      01JAN92   10JAN92    21JAN92   30JAN92       20         0
Facility       01JAN92   10JAN92    21JAN92   30JAN92       20         0
Init. Prod.    11JAN92   20JAN92    31JAN92   09FEB92       20         0
Evaluate       21JAN92   30JAN92    15FEB92   24FEB92       25         5
Test Market    21JAN92   04FEB92    10FEB92   24FEB92       20         0
Marketing      21JAN92   21JAN92    01MAR92   01MAR92       40        40
Dummy          21JAN92   21JAN92    10FEB92   10FEB92       20         0
Changes        05FEB92   09FEB92    25FEB92   29FEB92       20         0
Production     10FEB92   10FEB92    01MAR92   01MAR92       20        20
```

Example 2.4. 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 2.2 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 2.4.1 displays only the critical activities in the project, while the second calendar in Output 2.4.2 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 start in December.

```
proc cpm data=widgaoa out=save
   date='2dec91'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  */
proc calendar schedule
     data=crit;
   id e_start;
   where e_start <= '31dec91'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';
proc calendar schedule data=save;
   id e_start;
   where e_start <= '31dec91'd;
   var task;
   dur days;
run;
```

Output 2.4.1. Project Calendar: Critical Activities

```
                               Printing the Schedule on a Calendar
                                 Critical Activities in December

---------------------------------------------------------------------------------------------------
|                                                                                                 |
|                                          December  1991                                         |
|                                                                                                 |
|-------------------------------------------------------------------------------------------------|
|   Sunday    |   Monday    |   Tuesday   |  Wednesday  |  Thursday   |   Friday    |  Saturday   |
|-------------+-------------+-------------+-------------+-------------+-------------+-------------|
|      1      |      2      |      3      |      4      |      5      |      6      |      7      |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|             |+===========================Approve Plan============================+|+=Drawings==>|
|-------------+-------------+-------------+-------------+-------------+-------------+-------------|
|      8      |      9      |     10      |     11      |     12      |     13      |     14      |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|<===========================================Drawings============================================>|
|-------------+-------------+-------------+-------------+-------------+-------------+-------------|
|     15      |     16      |     17      |     18      |     19      |     20      |     21      |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|<========Drawings=========+|+=============================Prototype=============================>|
|-------------+-------------+-------------+-------------+-------------+-------------+-------------|
|     22      |     23      |     24      |     25      |     26      |     27      |     28      |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|<===========================================Prototype===========================================>|
|-------------+-------------+-------------+-------------+-------------+-------------+-------------|
|     29      |     30      |     31      |             |             |             |             |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|<===============Prototype===============+|             |             |             |             |
---------------------------------------------------------------------------------------------------
```

Output 2.4.2. Project Calendar: All Activities

```
                                 Printing the Schedule on a Calendar
                                  Early Start Schedule for December

---------------------------------------------------------------------------------------------------
|                                                                                                 |
|                                         December  1991                                          |
|                                                                                                 |
|-------------------------------------------------------------------------------------------------|
|   Sunday    |   Monday    |   Tuesday   |  Wednesday  |  Thursday   |   Friday    |  Saturday   |
|-------------+-------------+-------------+-------------+-------------+-------------+-------------|
|      1      |      2      |      3      |      4      |      5      |      6      |      7      |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |+Write Specs>|
|             |             |             |             |             |             |+Anal. Marke>|
|             |+===========================Approve Plan============================+|+=Drawings==>|
|-------------+-------------+-------------+-------------+-------------+-------------+-------------|
|      8      |      9      |     10      |     11      |     12      |     13      |     14      |
|             |             |             |             |             |             |             |
|<=====================Write Specs=====================+|             |             |             |
|<====================Anal. Market=====================+|+==============Mkt. Strat.==============>|
|<===========================================Drawings============================================>|
|-------------+-------------+-------------+-------------+-------------+-------------+-------------|
|     15      |     16      |     17      |     18      |     19      |     20      |     21      |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|<==========================================Mkt. Strat.==========================================+|
|<========Drawings=========+|+============================Prototype==============================>|
|-------------+-------------+-------------+-------------+-------------+-------------+-------------|
|     22      |     23      |     24      |     25      |     26      |     27      |     28      |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|<===========================================Prototype===========================================>|
|-------------+-------------+-------------+-------------+-------------+-------------+-------------|
|     29      |     30      |     31      |             |             |             |             |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|             |             |             |             |             |             |             |
|<===============Prototype==============+|             |             |             |             |
---------------------------------------------------------------------------------------------------
```

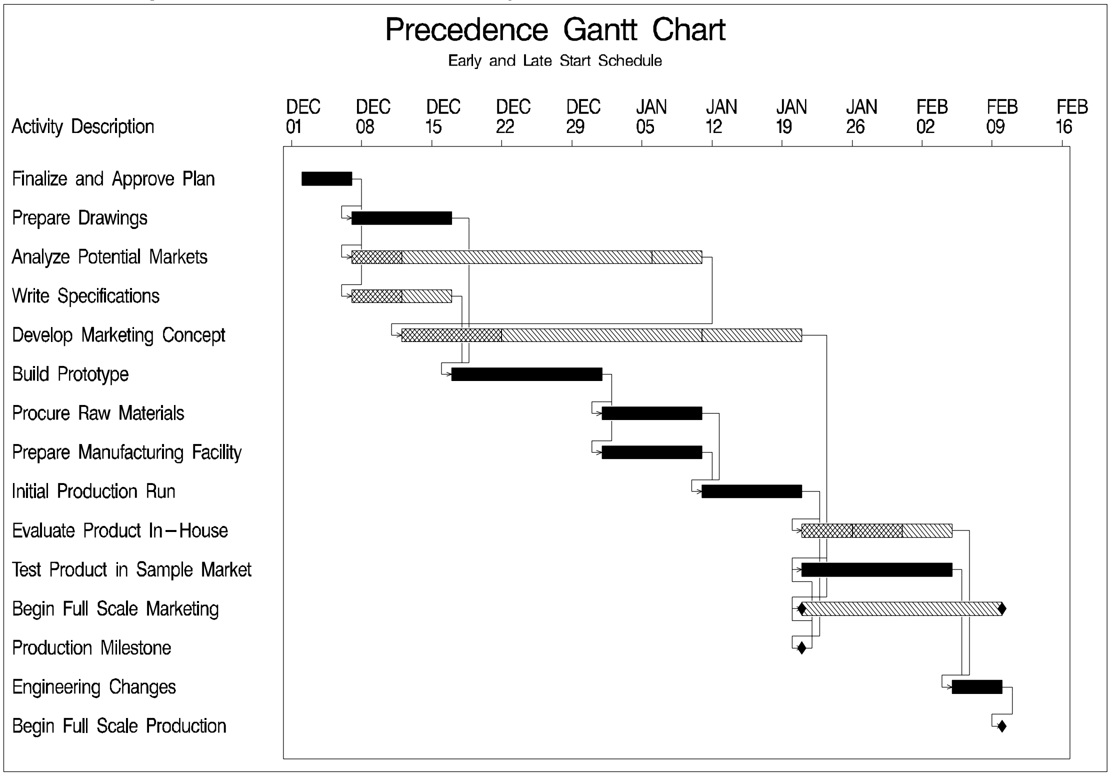
Example 2.5. Precedence Gantt Chart

This example produces a Gantt chart of the schedule obtained from PROC CPM. The example uses the network described in Example 2.2 (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 via the TAILNODE= and HEADNODE= options.

```
* specify the device on which you want the chart printed;

goptions vpos=50 hpos=80 border;

title f=swiss 'Precedence Gantt Chart';
title2 f=swiss 'Early and Late Start Schedule';

proc gantt graphics data=save;
   chart / compress tailnode=tail headnode=head
           font=swiss height=1.5 nojobnum skip=2
```

```
            cprec=cyan cmile=magenta
            caxis=black cframe=ligr
            dur=days increment=7 nolegend;
   id descrpt;
   run;
```

Output 2.5.1. Gantt Chart of Project



Example 2.6. 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 2.6.1 displays the schedule produced by PROC CPM. Note that, with a shorter work week, the project finishes on March 9, 1992, instead of on March 1, 1992.

```
proc cpm data=widget out=save
     date='2dec91'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 2.6.1. 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    02DEC91    06DEC91     02DEC91    06DEC91         0          0
Drawings        09DEC91    20DEC91     09DEC91    20DEC91         0          0
Anal. Market    09DEC91    13DEC91     20JAN92    24JAN92        30          0
Write Specs     09DEC91    13DEC91     16DEC91    20DEC91         5          5
Prototype       23DEC91    10JAN92     23DEC91    10JAN92         0          0
Mkt. Strat.     16DEC91    27DEC91     27JAN92    07FEB92        30         30
Materials       13JAN92    24JAN92     13JAN92    24JAN92         0          0
Facility        13JAN92    24JAN92     13JAN92    24JAN92         0          0
Init. Prod.     27JAN92    07FEB92     27JAN92    07FEB92         0          0
Evaluate        10FEB92    21FEB92     17FEB92    28FEB92         5          5
Test Market     10FEB92    28FEB92     10FEB92    28FEB92         0          0
Changes         02MAR92    06MAR92     02MAR92    06MAR92         0          0
Production      09MAR92    09MAR92     09MAR92    09MAR92         0          0
Marketing       10FEB92    10FEB92     09MAR92    09MAR92        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 2.6.2, 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('31dec91'd, e_finish);
   if e_start <= '31dec91'd;
   run;

title3 'Calendar of Schedule';
proc calendar data=december schedule weekdays;
   id e_start;
   finish e_finish;
   var task;
   run;
```

Output 2.6.2. Changing Duration Units: WEEKDAY Calendar for December

```
                          Changing Duration Units
                              INTERVAL=WEEKDAY
                            Calendar of Schedule

----------------------------------------------------------------------------
|                                                                          |
|                              December  1991                              |
|                                                                          |
|--------------------------------------------------------------------------|
|    Monday    |   Tuesday    |  Wednesday   |   Thursday   |    Friday    |
|--------------+--------------+--------------+--------------+--------------|
|       2      |       3      |       4      |       5      |       6      |
|              |              |              |              |              |
|              |              |              |              |              |
|              |              |              |              |              |
|              |              |              |              |              |
|+==============================Approve Plan==============================+|
|--------------+--------------+--------------+--------------+--------------|
|       9      |      10      |      11      |      12      |      13      |
|              |              |              |              |              |
|              |              |              |              |              |
|+==============================Write Specs===============================+|
|+==============================Anal. Market==============================+|
|+================================Drawings================================>|
|--------------+--------------+--------------+--------------+--------------|
|      16      |      17      |      18      |      19      |      20      |
|              |              |              |              |              |
|              |              |              |              |              |
|              |              |              |              |              |
|+==============================Mkt. Strat.===============================>|
|<================================Drawings================================+|
|--------------+--------------+--------------+--------------+--------------|
|      23      |      24      |      25      |      26      |      27      |
|              |              |              |              |              |
|              |              |              |              |              |
|              |              |              |              |              |
|+===============================Prototype================================>|
|<==============================Mkt. Strat.===============================+|
|--------------+--------------+--------------+--------------+--------------|
|      30      |      31      |              |              |              |
|              |              |              |              |              |
|              |              |              |              |              |
|              |              |              |              |              |
|              |              |              |              |              |
|<=========Prototype=========+|              |              |              |
----------------------------------------------------------------------------
```

Note that 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 2.6.3. 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='2dec91'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 2.6.3. 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	02DEC91	08DEC91	02DEC91	08DEC91	0	0
Drawings	09DEC91	22DEC91	09DEC91	22DEC91	0	0
Anal. Market	09DEC91	15DEC91	20JAN92	26JAN92	6	0
Write Specs	09DEC91	15DEC91	16DEC91	22DEC91	1	1
Prototype	23DEC91	12JAN92	23DEC91	12JAN92	0	0
Mkt. Strat.	16DEC91	29DEC91	27JAN92	09FEB92	6	6
Materials	13JAN92	26JAN92	13JAN92	26JAN92	0	0
Facility	13JAN92	26JAN92	13JAN92	26JAN92	0	0
Init. Prod.	27JAN92	09FEB92	27JAN92	09FEB92	0	0
Evaluate	10FEB92	23FEB92	17FEB92	01MAR92	1	1
Test Market	10FEB92	01MAR92	10FEB92	01MAR92	0	0
Changes	02MAR92	08MAR92	02MAR92	08MAR92	0	0
Production	09MAR92	09MAR92	09MAR92	09MAR92	0	0
Marketing	10FEB92	10FEB92	09MAR92	09MAR92	4	4

Example 2.7. Controlling the Project Calendar

This example illustrates the use of the INTERVAL=, DAYSTART=, and DAYLENGTH= options to control the project calendar. In Examples 2.1 through 2.5, 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 2.6, 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 2.1 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 2, with an eight-hour workday and a five-day work week, you can invoke PROC CPM with the following statements. Output 2.7.1 displays the resulting schedule.

```
title 'Controlling the Project Calendar';
title2 'Scheduling on Workdays';
proc cpm data=widget date='2dec91'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 2.7.1. 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

Approve Plan    02DEC91:09:00:00    06DEC91:16:59:59    02DEC91:09:00:00
Drawings        09DEC91:09:00:00    20DEC91:16:59:59    09DEC91:09:00:00
Anal. Market    09DEC91:09:00:00    13DEC91:16:59:59    20JAN92:09:00:00
Write Specs     09DEC91:09:00:00    13DEC91:16:59:59    16DEC91:09:00:00
Prototype       23DEC91:09:00:00    10JAN92:16:59:59    23DEC91:09:00:00
Mkt. Strat.     16DEC91:09:00:00    27DEC91:16:59:59    27JAN92:09:00:00
Materials       13JAN92:09:00:00    24JAN92:16:59:59    13JAN92:09:00:00
Facility        13JAN92:09:00:00    24JAN92:16:59:59    13JAN92:09:00:00
Init. Prod.     27JAN92:09:00:00    07FEB92:16:59:59    27JAN92:09:00:00
Evaluate        10FEB92:09:00:00    21FEB92:16:59:59    17FEB92:09:00:00
Test Market     10FEB92:09:00:00    28FEB92:16:59:59    10FEB92:09:00:00
Changes         02MAR92:09:00:00    06MAR92:16:59:59    02MAR92:09:00:00
Production      09MAR92:09:00:00    09MAR92:09:00:00    09MAR92:09:00:00
Marketing       10FEB92:09:00:00    10FEB92:09:00:00    09MAR92:09:00:00

task                   L_FINISH    T_FLOAT    F_FLOAT

Approve Plan    06DEC91:16:59:59        0          0
Drawings        20DEC91:16:59:59        0          0
Anal. Market    24JAN92:16:59:59       30          0
Write Specs     20DEC91:16:59:59        5          5
Prototype       10JAN92:16:59:59        0          0
Mkt. Strat.     07FEB92:16:59:59       30         30
Materials       24JAN92:16:59:59        0          0
Facility        24JAN92:16:59:59        0          0
Init. Prod.     07FEB92:16:59:59        0          0
Evaluate        28FEB92:16:59:59        5          5
Test Market     28FEB92:16:59:59        0          0
Changes         06MAR92:16:59:59        0          0
Production      09MAR92:09:00:00        0          0
Marketing       09MAR92:09:00:00       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 2. The project is scheduled on eight-and-a-half hour workdays each starting at 7 a.m. Output 2.7.2 displays the resulting schedule produced by PROC CPM.

```
proc cpm data=widget date='2dec91'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 2.7.2. 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

Approve Plan    02DEC91:07:00:00    06DEC91:15:29:59    02DEC91:07:00:00
Drawings        09DEC91:07:00:00    20DEC91:15:29:59    09DEC91:07:00:00
Anal. Market    09DEC91:07:00:00    13DEC91:15:29:59    20JAN92:07:00:00
Write Specs     09DEC91:07:00:00    13DEC91:15:29:59    16DEC91:07:00:00
Prototype       23DEC91:07:00:00    10JAN92:15:29:59    23DEC91:07:00:00
Mkt. Strat.     16DEC91:07:00:00    27DEC91:15:29:59    27JAN92:07:00:00
Materials       13JAN92:07:00:00    24JAN92:15:29:59    13JAN92:07:00:00
Facility        13JAN92:07:00:00    24JAN92:15:29:59    13JAN92:07:00:00
Init. Prod.     27JAN92:07:00:00    07FEB92:15:29:59    27JAN92:07:00:00
Evaluate        10FEB92:07:00:00    21FEB92:15:29:59    17FEB92:07:00:00
Test Market     10FEB92:07:00:00    28FEB92:15:29:59    10FEB92:07:00:00
Changes         02MAR92:07:00:00    06MAR92:15:29:59    02MAR92:07:00:00
Production      09MAR92:07:00:00    09MAR92:07:00:00    09MAR92:07:00:00
Marketing       10FEB92:07:00:00    10FEB92:07:00:00    09MAR92:07:00:00

task                     L_FINISH    T_FLOAT    F_FLOAT

Approve Plan    06DEC91:15:29:59        0           0
Drawings        20DEC91:15:29:59        0           0
Anal. Market    24JAN92:15:29:59       30           0
Write Specs     20DEC91:15:29:59        5           5
Prototype       10JAN92:15:29:59        0           0
Mkt. Strat.     07FEB92:15:29:59       30          30
Materials       24JAN92:15:29:59        0           0
Facility        24JAN92:15:29:59        0           0
Init. Prod.     07FEB92:15:29:59        0           0
Evaluate        28FEB92:15:29:59        5           5
Test Market     28FEB92:15:29:59        0           0
Changes         06MAR92:15:29:59        0           0
Production      09MAR92:07:00:00        0           0
Marketing       09MAR92:07:00:00       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='2dec91:07:00'dt in conjunction with the specification INTERVAL=DTWRKDAY and DAYLENGTH='08:30't.

Example 2.8. 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 2.8.1. Scheduling around Holidays: HOLIDAYS data set

```
                    Scheduling Around Holidays
                       Data Set HOLIDAYS

               Obs    holiday    holifin    holidur

                1     25DEC91    27DEC91       4
                2     01JAN92          .       .
```

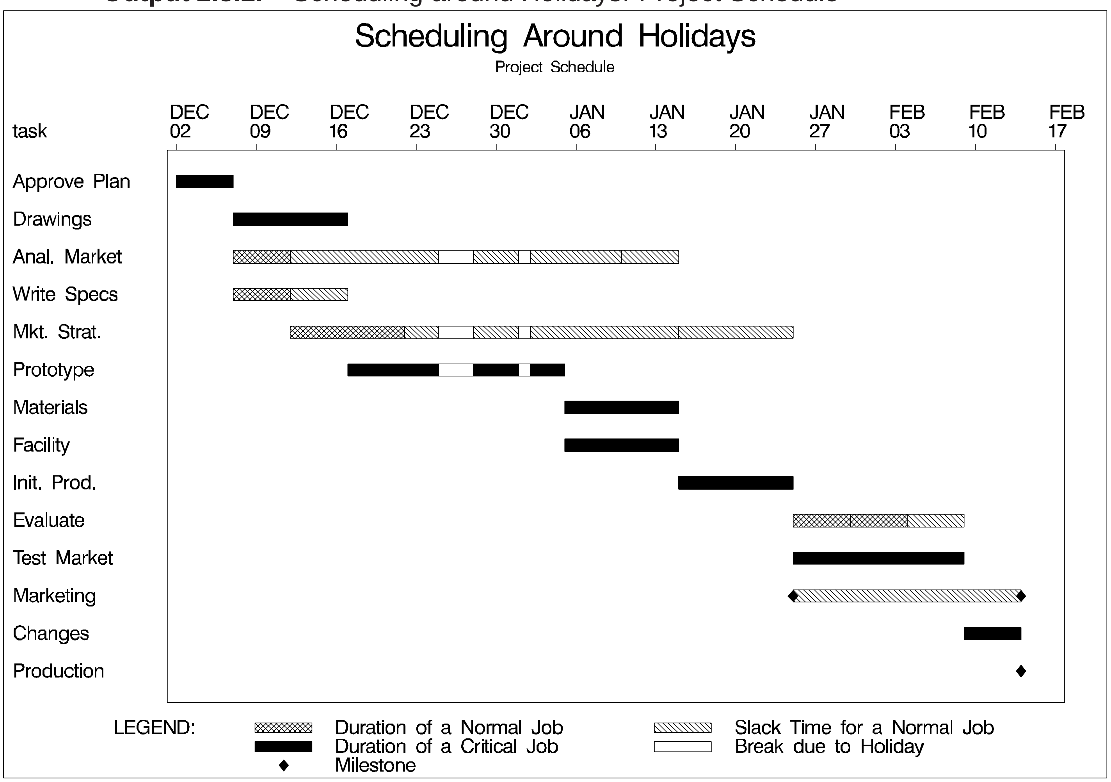
For example, the data set HOLIDAYS, displayed in Output 2.8.1 specifies two holidays, one for Christmas and the other for New Year's Day. The variable holiday specifies the start of each holiday. The variable holifin specifies the end of the Christmas holiday as 27Dec91. Alternately, the variable holidur can be used to interpret the Christmas holiday as lasting four interval units starting from the 25th 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 2.8.2 displays the schedule obtained.

```
proc cpm data=widget holidata=holidays
         out=saveh date='2dec91'd ;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   run;

proc sort data=saveh;
   by e_start;
   run;
```

```
title 'Scheduling Around Holidays';
title2 'Project Schedule';
goptions vpos=50 hpos=80 border;
goptions ftext=swiss;

proc gantt graphics data=saveh holidata=holidays;
   chart / compress
           font=swiss height=1.5 nojobnum skip=2
           dur=days increment=7
           holiday=(holiday) holifin=(holifin)
           cframe=ligr;
   id task;
   run;
```

Output 2.8.2. Scheduling around Holidays: Project Schedule

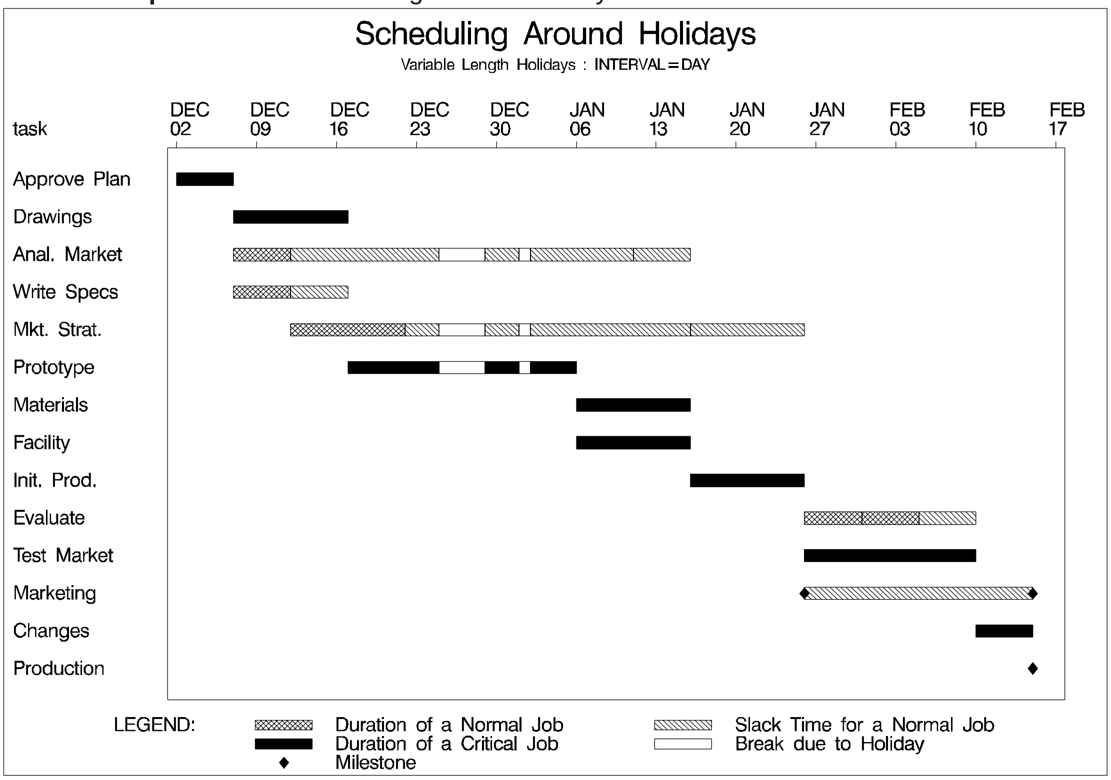


The next two invocations illustrate the use of the HOLIDUR= option and the effect of the INTERVAL= option on the duration of the holidays. Recall that the holiday duration is also assumed to be in *interval* units where *interval* is the value specified for the INTERVAL= option. Suppose that a holiday period for the entire project starts on December 25, 1991, with duration specified as 4. First the project is scheduled with INTERVAL=DAY so that the holidays are on December 25, 26, 27, and 28, 1991. Output 2.8.3 displays the resulting schedule. The project completion is delayed by one day due to the extra holiday on December 28, 1991.

```
proc cpm data=widget holidata=holidays
         out=saveh1 date='2dec91'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
              font=swiss
              height=1.5 skip=2
              nojobnum
              dur=days increment=7
              holiday=(holiday) holidur=(holidur) interval=day
              cframe=ligr;
      id task;
      run;
```

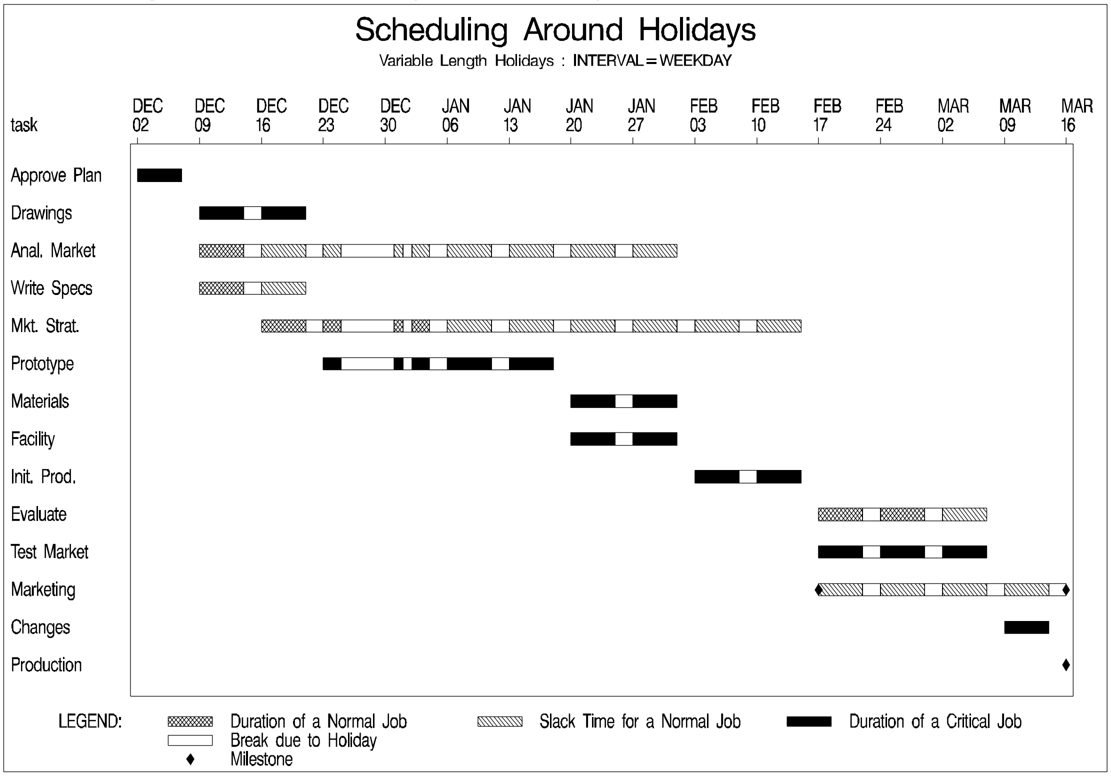
Output 2.8.3. Scheduling around Holidays: INTERVAL=DAY



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 25, 26, 27, and 30, 1991, because December 28 and 29 (Saturday and Sunday) are non-working days anyway. (Note that if holifin had been used, the holiday would have ended on December 27, 1991.) The following statements schedule the project to start on December 2, 1991 with INTERVAL=WEEKDAY. Output 2.8.4 displays the resulting schedule. Note the further delay in project completion time.

```
proc cpm data=widget holidata=holidays
         out=saveh2 date='2dec91'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
           font=swiss
           height=1.5 skip=2
           nojobnum
           dur=days increment=7
           holiday=(holiday)
           holidur=(holidur)
           interval=weekday
           cframe=ligr;
   id task;
   run;
```

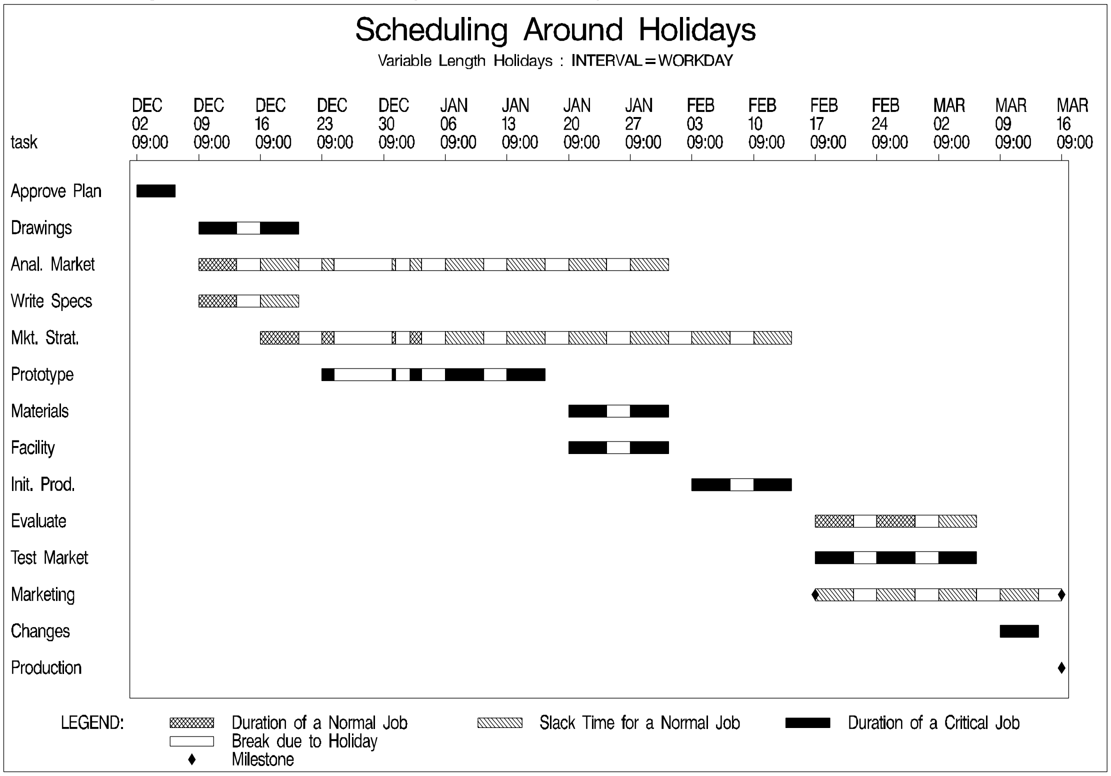
Output 2.8.4. Scheduling around Holidays: INTERVAL=WEEKDAY



Finally, the same project is scheduled to start on December 2, 1991 with INTERVAL=WORKDAY. Output 2.8.5 displays the resulting Schedule data set. Note that this time the holiday period starts at 5:00 p.m. on December 24, 1991, and ends at 9:00 a.m. on December 31, 1991.

```
proc cpm data=widget holidata=holidays
         out=saveh3 date='2dec91'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
           font=swiss height=1.5 nojobnum skip=2
           dur=days increment=7
           holiday=(holiday) holidur=(holidur) interval=workday
           cframe=ligr;
   id task;
   run;
```

Output 2.8.5. Scheduling around Holidays: INTERVAL=WORKDAY



Example 2.9. 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 2.8, but some of the durations have been changed to include some fractional values. Output 2.9.1 shows the project data set.

Output 2.9.1. 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	Anal. Market	Write Specs
2	Drawings	10.0	Prototype		
3	Anal. 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 2.9.2 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='2dec91:07:00'dt
         interval=dtday daylength='08:30't
         calendar=calendar;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   run;
```

Output 2.9.2. Scheduling on the Six-Day Week

Scheduling on the 6-Day Week
Project Schedule

Obs	task	days	E_START	E_FINISH
1	Approve Plan	5.5	02DEC91:07:00:00	07DEC91:11:14:59
2	Drawings	10.0	07DEC91:11:15:00	19DEC91:11:14:59
3	Anal. Market	5.0	07DEC91:11:15:00	13DEC91:11:14:59
4	Write Specs	4.5	07DEC91:11:15:00	12DEC91:15:29:59
5	Prototype	15.0	19DEC91:11:15:00	10JAN92:11:14:59
6	Mkt. Strat.	10.0	13DEC91:11:15:00	28DEC91:11:14:59
7	Materials	10.0	10JAN92:11:15:00	22JAN92:11:14:59
8	Facility	10.0	10JAN92:11:15:00	22JAN92:11:14:59
9	Init. Prod.	10.0	22JAN92:11:15:00	03FEB92:11:14:59
10	Evaluate	10.0	03FEB92:11:15:00	14FEB92:11:14:59
11	Test Market	15.0	03FEB92:11:15:00	20FEB92:11:14:59
12	Changes	5.0	20FEB92:11:15:00	26FEB92:11:14:59
13	Production	0.0	26FEB92:11:15:00	26FEB92:11:15:00
14	Marketing	0.0	03FEB92:11:15:00	03FEB92:11:15:00

Obs	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	02DEC91:07:00:00	07DEC91:11:14:59	0.0	0.0
2	07DEC91:11:15:00	19DEC91:11:14:59	0.0	0.0
3	16JAN92:11:15:00	22JAN92:11:14:59	30.0	0.0
4	14DEC91:07:00:00	19DEC91:11:14:59	5.5	5.5
5	19DEC91:11:15:00	10JAN92:11:14:59	0.0	0.0
6	22JAN92:11:15:00	03FEB92:11:14:59	30.0	30.0
7	10JAN92:11:15:00	22JAN92:11:14:59	0.0	0.0
8	10JAN92:11:15:00	22JAN92:11:14:59	0.0	0.0
9	22JAN92:11:15:00	03FEB92:11:14:59	0.0	0.0
10	08FEB92:11:15:00	20FEB92:11:14:59	5.0	5.0
11	03FEB92:11:15:00	20FEB92:11:14:59	0.0	0.0
12	20FEB92:11:15:00	26FEB92:11:14:59	0.0	0.0
13	26FEB92:11:15:00	26FEB92:11:15:00	0.0	0.0
14	26FEB92:11:15:00	26FEB92:11:15:00	20.0	20.0

Output 2.9.3. 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 2.9.4. 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
```

Output 2.9.5. 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

       1    Approve Plan     5.5    02DEC91:08:00:00    07DEC91:11:59:59
       2    Drawings        10.0    09DEC91:08:00:00    20DEC91:11:59:59
       3    Anal. Market     5.0    09DEC91:08:00:00    13DEC91:15:59:59
       4    Write Specs      4.5    09DEC91:08:00:00    13DEC91:11:59:59
       5    Prototype       15.0    20DEC91:12:00:00    14JAN92:11:59:59
       6    Mkt. Strat.     10.0    14DEC91:08:00:00    31DEC91:11:59:59
       7    Materials       10.0    14JAN92:12:00:00    27JAN92:11:59:59
       8    Facility        10.0    14JAN92:12:00:00    27JAN92:11:59:59
       9    Init. Prod.     10.0    27JAN92:12:00:00    07FEB92:15:59:59
      10    Evaluate        10.0    08FEB92:08:00:00    20FEB92:15:59:59
      11    Test Market     15.0    08FEB92:08:00:00    27FEB92:11:59:59
      12    Changes          5.0    27FEB92:12:00:00    04MAR92:15:59:59
      13    Production       0.0    05MAR92:08:00:00    05MAR92:08:00:00
      14    Marketing        0.0    08FEB92:08:00:00    08FEB92:08:00:00

     Obs             L_START            L_FINISH    T_FLOAT    F_FLOAT

       1    02DEC91:08:00:00    07DEC91:11:59:59       0.0        0.0
       2    09DEC91:08:00:00    20DEC91:11:59:59       0.0        0.0
       3    21JAN92:08:00:00    27JAN92:11:59:59      30.0        0.0
       4    16DEC91:08:00:00    20DEC91:11:59:59       5.5        5.5
       5    20DEC91:12:00:00    14JAN92:11:59:59       0.0        0.0
       6    27JAN92:12:00:00    07FEB92:15:59:59      30.0       30.0
       7    14JAN92:12:00:00    27JAN92:11:59:59       0.0        0.0
       8    14JAN92:12:00:00    27JAN92:11:59:59       0.0        0.0
       9    27JAN92:12:00:00    07FEB92:15:59:59       0.0        0.0
      10    14FEB92:12:00:00    27FEB92:11:59:59       5.0        5.0
      11    08FEB92:08:00:00    27FEB92:11:59:59       0.0        0.0
      12    27FEB92:12:00:00    04MAR92:15:59:59       0.0        0.0
      13    05MAR92:08:00:00    05MAR92:08:00:00       0.0        0.0
      14    05MAR92:08:00:00    05MAR92:08:00:00      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 2.9.3 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 2.9.4 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 2.9.5.

```
proc cpm data=widget9 holidata=holidays
         out=savecw date='2dec91'd
         interval=day
         workday=workdat calendar=caldat;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   run;
```

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 2.9.6. The period illustrated on the chart is from December 20, 1991 to December 28, 1991. 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='20dec91:08:00'dt;
   e_finish='28dec91:23:59:59'dt;
   task='Schedule Breaks';
   label task='Project Calendar';
   format e_start e_finish datetime16.;
   run;

options ps=20;
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 2.9.6. 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            20      20      21      21      21      22
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
  22      22      22      23      23      23      24      24
  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
  24      24      25      25      25      26      26      26
  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
  26      27      27      27      28      28      28      29
  16:00   00:00   08:00   16:00   00:00   08:00   16:00   00:00
 -+-------+-------+-------+-------+-------+-------+-------+-
 |****************************************----*************|
 -+-------+-------+-------+-------+-------+-------+-------+-
```

Example 2.10. 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 2.10.1 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 2.10.1. 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 2.9, 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 2.10.2 displays the Holiday data set.

Output 2.10.2. Holiday Data Set

Multiple Calendars
Holidays Data Set

Obs	holiday	holifin	holidur
1	25DEC91	27DEC91	4
2	01JAN92	01JAN92	.

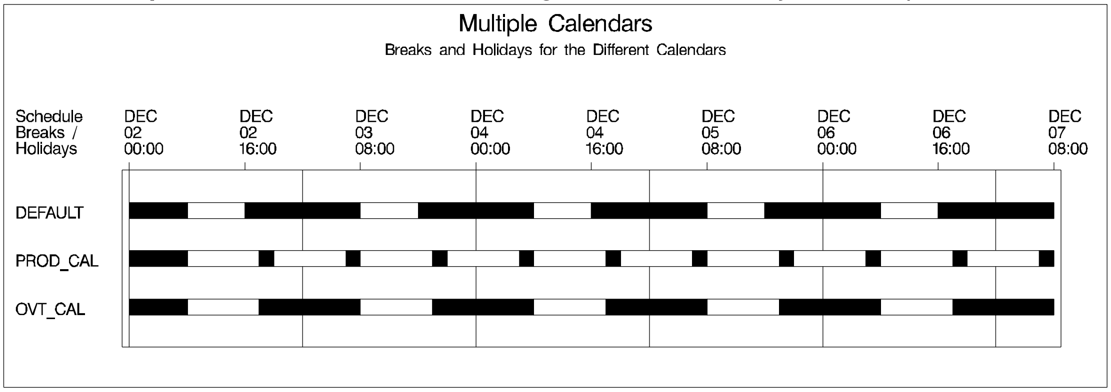
Note that the data set HOLIDAYS does not include any variable identifying the calendars with which to associate the holidays. By default, the procedure associates the two holiday periods with all the calendars.

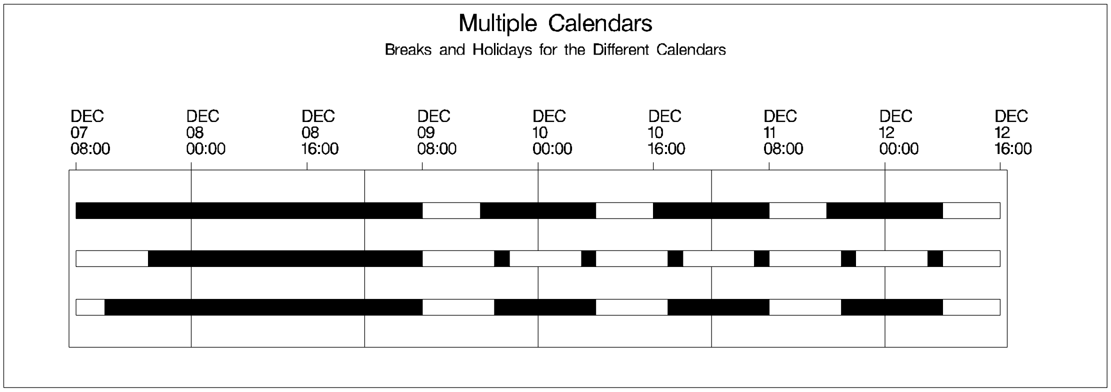
An easy way to visualize all the breaks and holidays for each calendar is to use a Gantt chart, plotting a bar for each calendar from the start of the project to January

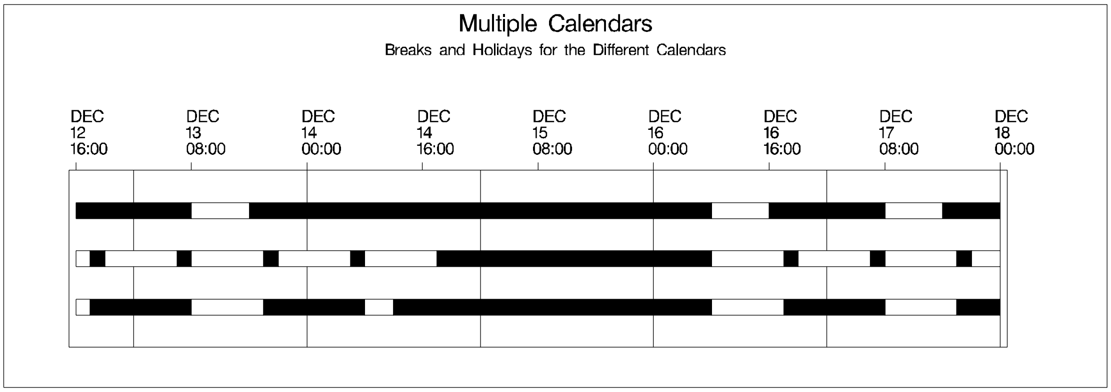
2, 1992, with all the holiday and work shift specifications. The following program produces Output 2.10.3. Note that holidays and breaks are marked with a solid fill pattern.

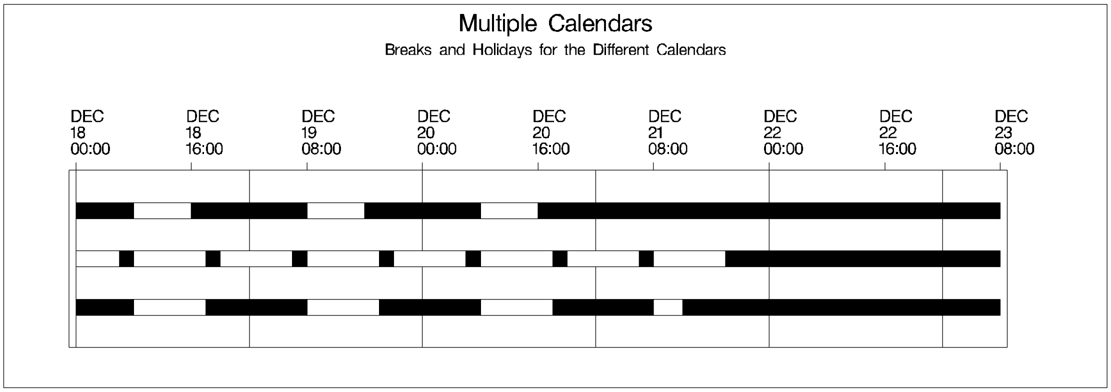
```
goptions hpos=160 vpos=25 ftext=swiss;
title h=1.5 'Multiple Calendars';
title2 'Breaks and Holidays for the Different Calendars';
proc gantt data=cals graphics
           calendar=calendar holidata=holidays
           workday=workdata;
   chart / interval=dtday mininterval=dthour skip=2
           holiday=(holiday) holifin=(holifin)
           markbreak daylength='08:00't calid=cal
           ref='2dec91:00:00'dt to '2jan92:00:00'dt by dtday
           nolegend nojobnum increment=16
           hpages=6;
   id cal;
   run;
```

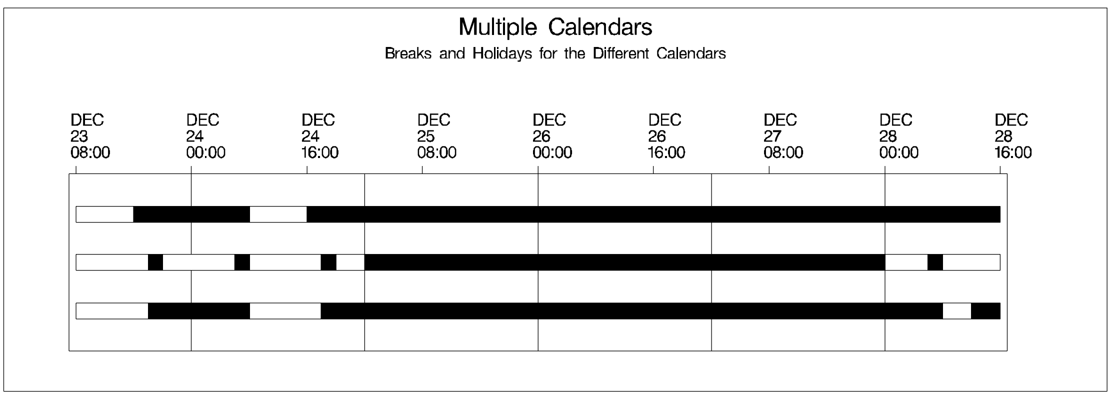
Output 2.10.3. Gantt Chart Showing Breaks and Holidays for Multiple Calendars

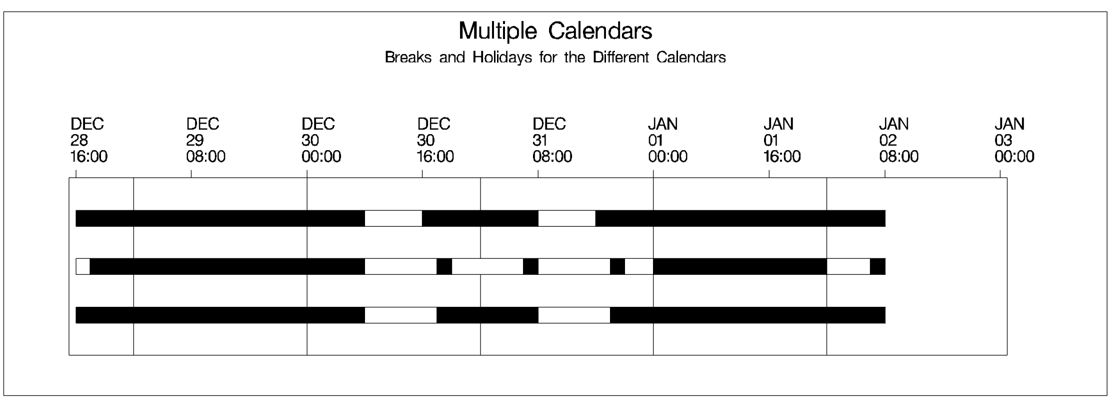





Multiple Calendars
Breaks and Holidays for the Different Calendars
DEC 12 16:00
DEC 13 08:00
DEC 14 00:00
DEC 14 16:00
DEC 15 08:00
DEC 16 00:00
DEC 16 16:00
DEC 17 08:00
DEC 18 00:00


Multiple Calendars
Breaks and Holidays for the Different Calendars
DEC 18 00:00
DEC 18 16:00
DEC 19 08:00
DEC 20 00:00
DEC 20 16:00
DEC 21 08:00
DEC 22 00:00
DEC 22 16:00
DEC 23 08:00


Multiple Calendars
Breaks and Holidays for the Different Calendars
DEC 23 08:00
DEC 24 00:00
DEC 24 16:00
DEC 25 08:00
DEC 26 00:00
DEC 26 16:00
DEC 27 08:00
DEC 28 00:00
DEC 28 16:00


Multiple Calendars
Breaks and Holidays for the Different Calendars
DEC 28 16:00
DEC 29 08:00
DEC 30 00:00
DEC 30 16:00
DEC 31 08:00
JAN 01 00:00
JAN 01 16:00
JAN 02 08:00
JAN 03 00:00

The Activity data set used in Example 2.9 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 issued 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 2.10.4 displays the schedule obtained. Note that the project is scheduled to finish on March 13, 1992, 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='02dec91'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;
   var task days e_start e_finish l_start l_finish
       t_float f_float;
   run;
```

Output 2.10.4. Schedule using Default Calendar

```
                          Multiple Calendars
                  Project Schedule: Default calendar

Obs    task            days              E_START            E_FINISH

  1    Approve Plan     5.5    02DEC91:08:00:00    09DEC91:11:59:59
  2    Drawings        10.0    09DEC91:12:00:00    23DEC91:11:59:59
  3    Anal. Market     5.0    09DEC91:12:00:00    16DEC91:11:59:59
  4    Write Specs      4.5    09DEC91:12:00:00    13DEC91:15:59:59
  5    Prototype       15.0    23DEC91:12:00:00    17JAN92:11:59:59
  6    Mkt. Strat.     10.0    16DEC91:12:00:00    03JAN92:11:59:59
  7    Materials       10.0    17JAN92:12:00:00    31JAN92:11:59:59
  8    Facility        10.0    17JAN92:12:00:00    31JAN92:11:59:59
  9    Init. Prod.     10.0    31JAN92:12:00:00    14FEB92:11:59:59
 10    Evaluate        10.0    14FEB92:12:00:00    28FEB92:11:59:59
 11    Test Market     15.0    14FEB92:12:00:00    06MAR92:11:59:59
 12    Changes          5.0    06MAR92:12:00:00    13MAR92:11:59:59
 13    Production       0.0    13MAR92:12:00:00    13MAR92:12:00:00
 14    Marketing        0.0    14FEB92:12:00:00    14FEB92:12:00:00

Obs             L_START            L_FINISH    T_FLOAT    F_FLOAT

  1    02DEC91:08:00:00    09DEC91:11:59:59       0.0        0.0
  2    09DEC91:12:00:00    23DEC91:11:59:59       0.0        0.0
  3    24JAN92:12:00:00    31JAN92:11:59:59      30.0        0.0
  4    17DEC91:08:00:00    23DEC91:11:59:59       5.5        5.5
  5    23DEC91:12:00:00    17JAN92:11:59:59       0.0        0.0
  6    31JAN92:12:00:00    14FEB92:11:59:59      30.0       30.0
  7    17JAN92:12:00:00    31JAN92:11:59:59       0.0        0.0
  8    17JAN92:12:00:00    31JAN92:11:59:59       0.0        0.0
  9    31JAN92:12:00:00    14FEB92:11:59:59       0.0        0.0
 10    21FEB92:12:00:00    06MAR92:11:59:59       5.0        5.0
 11    14FEB92:12:00:00    06MAR92:11:59:59       0.0        0.0
 12    06MAR92:12:00:00    13MAR92:11:59:59       0.0        0.0
 13    13MAR92:12:00:00    13MAR92:12:00:00       0.0        0.0
 14    13MAR92:12:00:00    13MAR92:12:00:00      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 2.10.5 shows that, due to longer working hours for these two activities in the project, the scheduled finish date is now March 9, at 10:00 a.m.

```
   proc cpm date='02dec91'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 2.10.5. Schedule using Three Calendars

```
                               Multiple Calendars
                       Project Schedule: Three Calendars

Obs    task            days      cal              E_START           E_FINISH

  1    Approve Plan     5.5    DEFAULT     02DEC91:08:00:00    09DEC91:11:59:59
  2    Drawings        10.0    DEFAULT     09DEC91:12:00:00    23DEC91:11:59:59
  3    Anal. Market     5.0    DEFAULT     09DEC91:12:00:00    16DEC91:11:59:59
  4    Write Specs      4.5    DEFAULT     09DEC91:12:00:00    13DEC91:15:59:59
  5    Prototype       15.0    OVT_CAL     23DEC91:12:00:00    13JAN92:09:59:59
  6    Mkt. Strat.     10.0    DEFAULT     16DEC91:12:00:00    03JAN92:11:59:59
  7    Materials       10.0    DEFAULT     13JAN92:10:00:00    27JAN92:09:59:59
  8    Facility        10.0    DEFAULT     13JAN92:10:00:00    27JAN92:09:59:59
  9    Init. Prod.     10.0    DEFAULT     27JAN92:10:00:00    10FEB92:09:59:59
 10    Evaluate        10.0    DEFAULT     10FEB92:10:00:00    24FEB92:09:59:59
 11    Test Market     15.0    DEFAULT     10FEB92:10:00:00    02MAR92:09:59:59
 12    Changes          5.0    DEFAULT     02MAR92:10:00:00    09MAR92:09:59:59
 13    Production       0.0    PROD_CAL    09MAR92:10:00:00    09MAR92:10:00:00
 14    Marketing        0.0    DEFAULT     10FEB92:10:00:00    10FEB92:10:00:00

Obs             L_START           L_FINISH    T_FLOAT    F_FLOAT

  1    02DEC91:08:00:00    09DEC91:11:59:59     0.00       0.00
  2    09DEC91:12:00:00    23DEC91:11:59:59     0.00       0.00
  3    20JAN92:10:00:00    27JAN92:09:59:59    25.75       0.00
  4    17DEC91:08:00:00    23DEC91:11:59:59     5.50       5.50
  5    23DEC91:12:00:00    13JAN92:09:59:59     0.00       0.00
  6    27JAN92:10:00:00    10FEB92:09:59:59    25.75      25.75
  7    13JAN92:10:00:00    27JAN92:09:59:59     0.00       0.00
  8    13JAN92:10:00:00    27JAN92:09:59:59     0.00       0.00
  9    27JAN92:10:00:00    10FEB92:09:59:59     0.00       0.00
 10    17FEB92:10:00:00    02MAR92:09:59:59     5.00       5.00
 11    10FEB92:10:00:00    02MAR92:09:59:59     0.00       0.00
 12    02MAR92:10:00:00    09MAR92:09:59:59     0.00       0.00
 13    09MAR92:10:00:00    09MAR92:10:00:00     0.00       0.00
 14    09MAR92:10:00:00    09MAR92:10:00:00    20.00      20.00
```

Now suppose that the engineer in charge of writing specifications requests a seven-day vacation from December 9, 1991. 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 2.10.6 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.

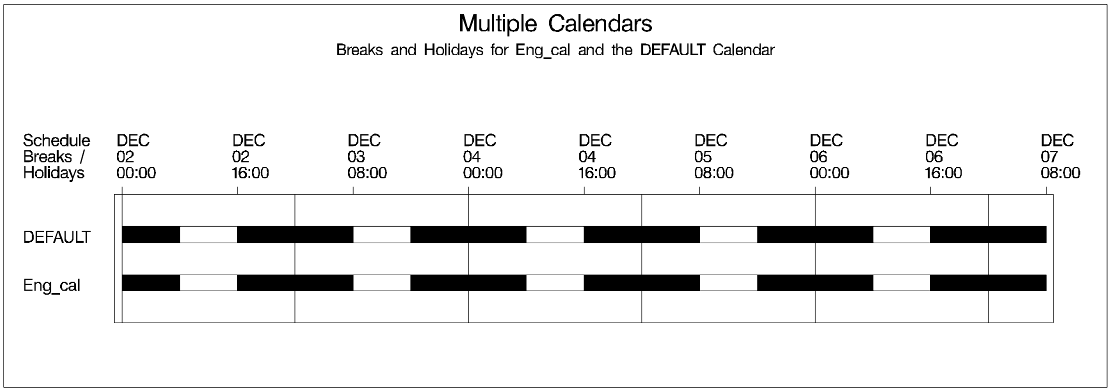
Output 2.10.6. HOLIDATA and CALEDATA Data Sets

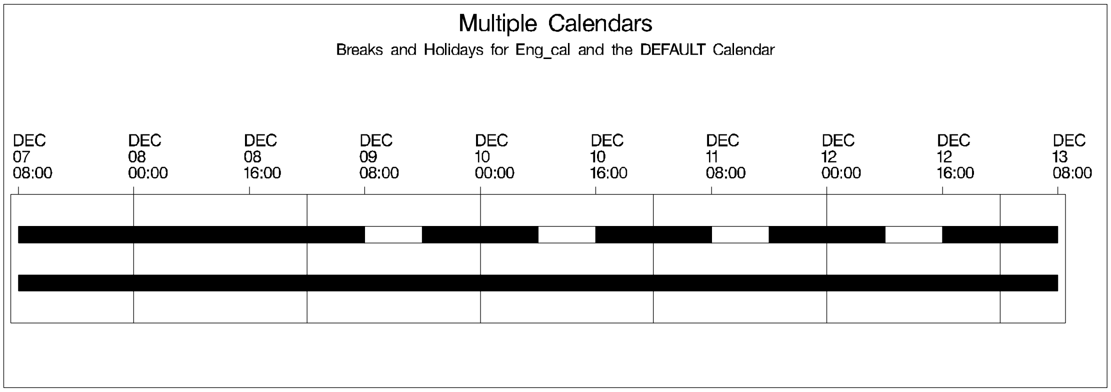
```
                         Multiple Calendars
                         Holidays Data Set

             Obs    holiday    holifin    holidur      cal

              1     09DEC91          .       7       Eng_cal
              2     25DEC91    27DEC91       .
              3     01JAN92    01JAN92       .



                         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
```

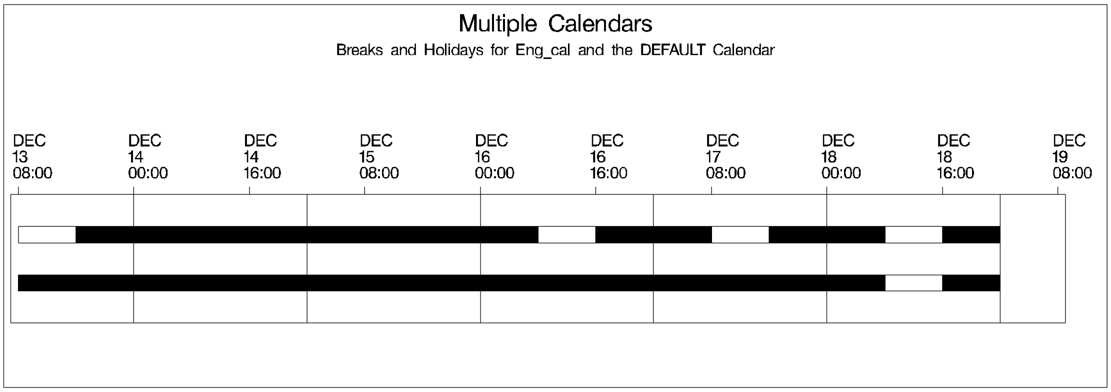
Once again, in the following code, PROC GANTT is used to compare the new calendar with the default calendar, as shown in Output 2.10.7. Note that the breaks and holidays are marked with a solid fill pattern.

```
/* Create a data set to illustrate holidays with PROC GANTT */
data cals2;
   e_start='2dec91:00:00'dt;
   e_finish='19dec91:00:00'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='2dec91:00:00'dt to '19dec91:00:00'dt by dtday
           nojobnum nolegend increment=16 hpages=3;
   id cal;
   run;
```

Output 2.10.7. 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 2.10.8 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 10, 1992; 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='02dec91'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 2.10.8. Schedule Using Four Calendars

Multiple Calendars
Project Schedule: Four Calendars

Obs	task	days	cal	E_START	E_FINISH
1	Approve Plan	5.5	DEFAULT	02DEC91:08:00:00	09DEC91:11:59:59
2	Drawings	10.0	DEFAULT	09DEC91:12:00:00	23DEC91:11:59:59
3	Anal. Market	5.0	DEFAULT	09DEC91:12:00:00	16DEC91:11:59:59
4	Write Specs	4.5	Eng_cal	18DEC91:08:00:00	24DEC91:11:59:59
5	Prototype	15.0	OVT_CAL	24DEC91:12:00:00	14JAN92:09:59:59
6	Mkt. Strat.	10.0	DEFAULT	16DEC91:12:00:00	03JAN92:11:59:59
7	Materials	10.0	DEFAULT	14JAN92:10:00:00	28JAN92:09:59:59
8	Facility	10.0	DEFAULT	14JAN92:10:00:00	28JAN92:09:59:59
9	Init. Prod.	10.0	DEFAULT	28JAN92:10:00:00	11FEB92:09:59:59
10	Evaluate	10.0	DEFAULT	11FEB92:10:00:00	25FEB92:09:59:59
11	Test Market	15.0	DEFAULT	11FEB92:10:00:00	03MAR92:09:59:59
12	Changes	5.0	DEFAULT	03MAR92:10:00:00	10MAR92:09:59:59
13	Production	0.0	PROD_CAL	10MAR92:10:00:00	10MAR92:10:00:00
14	Marketing	0.0	DEFAULT	11FEB92:10:00:00	11FEB92:10:00:00

Obs	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	03DEC91:08:00:00	10DEC91:11:59:59	1.00	0.00
2	10DEC91:12:00:00	24DEC91:11:59:59	1.00	1.00
3	21JAN92:10:00:00	28JAN92:09:59:59	26.75	0.00
4	18DEC91:08:00:00	24DEC91:11:59:59	0.00	0.00
5	24DEC91:12:00:00	14JAN92:09:59:59	0.00	0.00
6	28JAN92:10:00:00	11FEB92:09:59:59	26.75	26.75
7	14JAN92:10:00:00	28JAN92:09:59:59	0.00	0.00
8	14JAN92:10:00:00	28JAN92:09:59:59	0.00	0.00
9	28JAN92:10:00:00	11FEB92:09:59:59	0.00	0.00
10	18FEB92:10:00:00	03MAR92:09:59:59	5.00	5.00
11	11FEB92:10:00:00	03MAR92:09:59:59	0.00	0.00
12	03MAR92:10:00:00	10MAR92:09:59:59	0.00	0.00
13	10MAR92:10:00:00	10MAR92:10:00:00	0.00	0.00
14	10MAR92:10:00:00	10MAR92:10:00:00	20.00	20.00

Example 2.11. Nonstandard Relationships

This example shows the use of LAG variables to describe nonstandard relationships. Consider the project network in AON format. Output 2.11.1 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. Note that 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 2.11.1. Network Data

Non-Standard Relationships
Activity Data Set WIDGLAG

Obs	task	days	succ	lagdur	lagdurc
1	Approve Plan	5	Drawings		
2	Approve Plan	5	Anal. Market		
3	Approve Plan	5	Write Specs		
4	Drawings	10	Prototype		
5	Anal. 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 2.6 that the project finishes on March 9, 1992. 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 2.11.2) follow.

```
proc cpm data=widglag date='2dec91'd
         interval=weekday collapse out=lagsched;
   activity task;
   succ     succ / lag = (lagdur);
   duration days;
   run;
```

Output 2.11.2. 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	02DEC91	06DEC91	02DEC91	06DEC91	0	0
Drawings	09DEC91	20DEC91	09DEC91	20DEC91	0	0
Anal. Market	09DEC91	13DEC91	14JAN92	20JAN92	26	0
Write Specs	09DEC91	13DEC91	16DEC91	20DEC91	5	5
Prototype	23DEC91	10JAN92	23DEC91	10JAN92	0	0
Mkt. Strat.	16DEC91	27DEC91	21JAN92	03FEB92	26	26
Materials	03JAN92	16JAN92	07JAN92	20JAN92	2	2
Facility	03JAN92	16JAN92	03JAN92	16JAN92	0	0
Init. Prod.	21JAN92	03FEB92	21JAN92	03FEB92	0	0
Evaluate	04FEB92	17FEB92	11FEB92	24FEB92	5	5
Test Market	04FEB92	24FEB92	04FEB92	24FEB92	0	0
Changes	25FEB92	02MAR92	25FEB92	02MAR92	0	0
Production	03MAR92	03MAR92	03MAR92	03MAR92	0	0
Marketing	04FEB92	04FEB92	03MAR92	03MAR92	20	20

Note that due to the change in the type of precedence constraint, the project finishes earlier, on March 3, 1992, instead of on March 9, 1992 (compare with Output 2.6.1).

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 2.11.3. PROC CPM is invoked with LAG=lagdurc and Output 2.11.4 displays the resulting schedule. Note that the project now finishes on March 2, 1992.

```
proc cpm data=widglag date='2dec91'd calendar=calendar
         interval=weekday collapse out=lagsched;
   activity task;
   succ     succ / lag = (lagdurc);
   duration days;
   run;
```

Output 2.11.3. 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
```

Output 2.11.4. 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    02DEC91    06DEC91     03DEC91    09DEC91        1          0
Drawings        09DEC91    20DEC91     10DEC91    23DEC91        1          0
Anal. Market    09DEC91    13DEC91     13JAN92    17JAN92       25          0
Write Specs     09DEC91    13DEC91     17DEC91    23DEC91        6          5
Prototype       23DEC91    10JAN92     24DEC91    13JAN92        1          0
Mkt. Strat.     16DEC91    27DEC91     20JAN92    31JAN92       25         25
Materials       03JAN92    16JAN92     06JAN92    17JAN92        1          1
Facility        03JAN92    16JAN92     06JAN92    17JAN92        1          1
Init. Prod.     20JAN92    31JAN92     20JAN92    31JAN92        0          0
Evaluate        03FEB92    14FEB92     10FEB92    21FEB92        5          5
Test Market     03FEB92    21FEB92     03FEB92    21FEB92        0          0
Changes         24FEB92    28FEB92     24FEB92    28FEB92        0          0
Production      02MAR92    02MAR92     02MAR92    02MAR92        0          0
Marketing       03FEB92    03FEB92     02MAR92    02MAR92       20         20
```

In fact, you can specify an alternate calendar for *all* the lag durations by using the ALAGCAL= or NLAGCAL= option in the SUCCESOR 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 2.11.5 shows the resulting schedule. Note that now the project finishes on February 28, 1992. Output 2.11.6 displays a precedence Gantt chart of the project. Note how the nonstandard precedence constraints are displayed.

```
proc cpm data=widglag date='2dec91'd calendar=calendar
         interval=weekday collapse out=lagsched;
   activity task;
   succ     succ / lag = (lagdur) alagcal=sevenday;
   duration days;
   run;

goptions hpos=100 vpos=60;
title  c=black f=swiss h=2.5 'Non-Standard Relationships';
title2 c=black f=swiss h=2   'Precedence Gantt Chart';
title3 ' ';
```

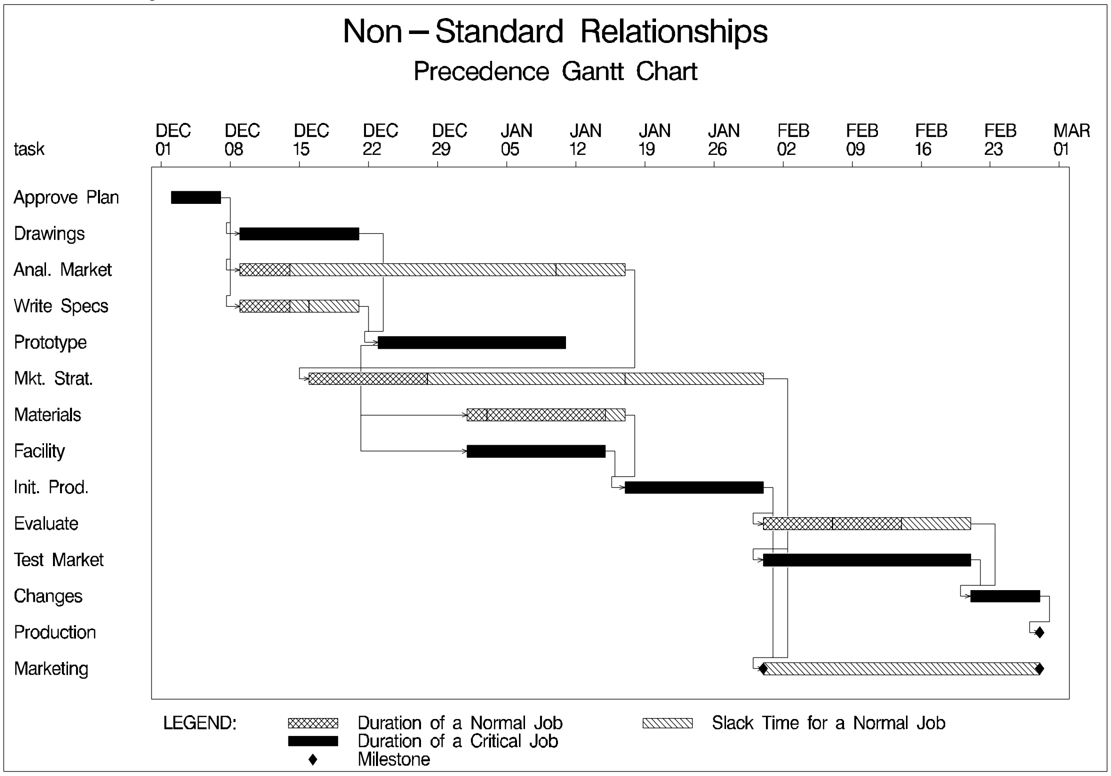
```
proc gantt graphics data=lagsched logic=widglag;
   chart / compress act=task succ=(succ) dur=days
           font=swiss
           cprec=black cmile=blue
           caxis=black cfram=cyan
           height=1.5 skip=2 nojobnum
           dur=days increment=7 lag=(lagdur);
   id task;
   run;
```

Output 2.11.5. 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	02DEC91	06DEC91	02DEC91	06DEC91	0	0
Drawings	09DEC91	20DEC91	09DEC91	20DEC91	0	0
Anal. Market	09DEC91	13DEC91	10JAN92	16JAN92	24	0
Write Specs	09DEC91	13DEC91	16DEC91	20DEC91	5	5
Prototype	23DEC91	10JAN92	23DEC91	10JAN92	0	0
Mkt. Strat.	16DEC91	27DEC91	17JAN92	30JAN92	24	24
Materials	01JAN92	14JAN92	03JAN92	16JAN92	2	2
Facility	01JAN92	14JAN92	01JAN92	14JAN92	0	0
Init. Prod.	17JAN92	30JAN92	17JAN92	30JAN92	0	0
Evaluate	31JAN92	13FEB92	07FEB92	20FEB92	5	5
Test Market	31JAN92	20FEB92	31JAN92	20FEB92	0	0
Changes	21FEB92	27FEB92	21FEB92	27FEB92	0	0
Production	28FEB92	28FEB92	28FEB92	28FEB92	0	0
Marketing	31JAN92	31JAN92	28FEB92	28FEB92	20	20

Output 2.11.6. Precedence Gantt Chart



Example 2.12. 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 2.12.1 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 by the 16th of December. The activity 'Test Market' has a *mandatory* start date imposed on it.

Output 2.12.1. Activity Data Set WIDGET12

Activity Time Constraints
Activity data set

Obs	task	days	succ1	succ2	succ3	adate	atype
1	Approve Plan	5	Drawings	Anal. Market	Write Specs	.	
2	Drawings	10	Prototype			16DEC91	feq
3	Anal. Market	5	Mkt. Strat.			.	
4	Write Specs	5	Prototype			16DEC91	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			28FEB92	fle
11	Test Market	15	Changes			17FEB92	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 2.12.2 shows the resulting schedule.

```
proc cpm data=widget12 date='2dec91'd
     xfervars interval=weekday;
   activity task;
   successor succ1 succ2 succ3;
   duration days;
   aligndate adate;
   aligntype atype;
   run;

options ls=90;
title 'Activity Time Constraints';
title2 'Aligned Schedule';
proc print;
   id task;
   var adate atype e_: l_: t_float f_float;
   run;
```

Output 2.12.2. Aligned Schedule

```
                              Activity Time Constraints
                                   Aligned Schedule

task             adate   atype  E_START  E_FINISH  L_START  L_FINISH  T_FLOAT  F_FLOAT

Approve Plan         .          02DEC91  06DEC91   26NOV91  02DEC91      -4       -4
Drawings      16DEC91   feq     09DEC91  20DEC91   03DEC91  16DEC91      -4       -4
Anal. Market         .          09DEC91  13DEC91   27JAN92  31JAN92      35        0
Write Specs   16DEC91   sge     16DEC91  20DEC91   23DEC91  27DEC91       5        0
Prototype            .          23DEC91  10JAN92   30DEC91  17JAN92       5        0
Mkt. Strat.          .          16DEC91  27DEC91   03FEB92  14FEB92      35       30
Materials            .          13JAN92  24JAN92   20JAN92  31JAN92       5        0
Facility             .          13JAN92  24JAN92   20JAN92  31JAN92       5        0
Init. Prod.          .          27JAN92  07FEB92   03FEB92  14FEB92       5        0
Evaluate      28FEB92   fle     10FEB92  21FEB92   17FEB92  28FEB92       5        5
Test Market   17FEB92   ms      17FEB92  06MAR92   17FEB92  06MAR92       0        0
Changes              .          09MAR92  13MAR92   09MAR92  13MAR92       0        0
Production           .          16MAR92  16MAR92   16MAR92  16MAR92       0        0
Marketing            .          10FEB92  10FEB92   16MAR92  16MAR92      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 17, 1992. However, the other types of constraints are not mandatory; they are superceded 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 16th of December (by the ‘fle’ constraint), the early start schedule causes it to finish on the 20th 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 2.13. 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 2.1, for the network in AON format. Suppose that the project has started as scheduled on December 2, 1991, and that the current date is December 20, 1991. 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 2.13.1 displays the resulting output data set.

```
data holidays;
   format holiday holifin date7.;
   input holiday date8. holifin date8. holidur;
   datalines;
25dec91 27dec91 4
01jan92 .       .
;

* store early schedule as the baseline schedule;

proc cpm data=widget holidata=holidays
         out=widgbase date='2dec91'd;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   baseline / set=early;
   run;
```

Output 2.13.1. Target Schedule

Progress Update and Target Schedules
Set Baseline Schedule

Obs	task	succ1	succ2	succ3	days	E_START
1	Approve Plan	Drawings	Anal. Market	Write Specs	5	02DEC91
2	Drawings	Prototype			10	07DEC91
3	Anal. Market	Mkt. Strat.			5	07DEC91
4	Write Specs	Prototype			5	07DEC91
5	Prototype	Materials	Facility		15	17DEC91
6	Mkt. Strat.	Test Market	Marketing		10	12DEC91
7	Materials	Init. Prod.			10	05JAN92
8	Facility	Init. Prod.			10	05JAN92
9	Init. Prod.	Test Market	Marketing	Evaluate	10	15JAN92
10	Evaluate	Changes			10	25JAN92
11	Test Market	Changes			15	25JAN92
12	Changes	Production			5	09FEB92
13	Production				0	14FEB92
14	Marketing				0	25JAN92

Obs	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT	B_START	B_FINISH
1	06DEC91	02DEC91	06DEC91	0	0	02DEC91	06DEC91
2	16DEC91	07DEC91	16DEC91	0	0	07DEC91	16DEC91
3	11DEC91	10JAN92	14JAN92	30	0	07DEC91	11DEC91
4	11DEC91	12DEC91	16DEC91	5	5	07DEC91	11DEC91
5	04JAN92	17DEC91	04JAN92	0	0	17DEC91	04JAN92
6	21DEC91	15JAN92	24JAN92	30	30	12DEC91	21DEC91
7	14JAN92	05JAN92	14JAN92	0	0	05JAN92	14JAN92
8	14JAN92	05JAN92	14JAN92	0	0	05JAN92	14JAN92
9	24JAN92	15JAN92	24JAN92	0	0	15JAN92	24JAN92
10	03FEB92	30JAN92	08FEB92	5	5	25JAN92	03FEB92
11	08FEB92	25JAN92	08FEB92	0	0	25JAN92	08FEB92
12	13FEB92	09FEB92	13FEB92	0	0	09FEB92	13FEB92
13	14FEB92	14FEB92	14FEB92	0	0	14FEB92	14FEB92
14	25JAN92	14FEB92	14FEB92	20	20	25JAN92	25JAN92

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 2.13.2. 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 2.13.2. Progress Data Set ACTUAL

```
                 Progress Update and Target Schedules
                            Progress Data

    Obs    task               sdate        fdate    pctc    rdur

      1    Approve Plan    02DEC1991    06DEC1991      .       .
      2    Drawings        07DEC1991    17DEC1991      .       .
      3    Anal. Market    06DEC1991            .    100       .
      4    Write Specs     08DEC1991    13DEC1991      .       .
      5    Prototype               .            .      .       .
      6    Mkt. Strat.     11DEC1991            .      .       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 2.13.3. Note that the activity ‘Mkt. Strat.’, which has rdur=‘3’ in Output 2.13.2, has an early finish time (December 22, 1992) 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='2dec91'd;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   baseline / compare=early;
   actual / a_start=sdate a_finish=fdate timenow='20dec91'd
            remdur=rdur pctcomp=pctc noautoupdt;
   run;
```

Output 2.13.3. Comparison of Schedules: NOAUTOUPDT

```
                    Progress Update and Target Schedules
              Updated Schedule vs. Target Schedule: NOAUTOUPDT

Obs task          succ1        succ2           succ3     days  STATUS

  1 Approve Plan  Drawings     Anal. Market  Write Specs    5  Completed
  2 Drawings      Prototype                                10  Completed
  3 Anal. Market  Mkt. Strat.                               5  Completed
  4 Write Specs   Prototype                                 5  Completed
  5 Prototype     Materials    Facility                    15  Pending
  6 Mkt. Strat.   Test Market  Marketing                   10  In Progress
  7 Materials     Init. Prod.                              10  Pending
  8 Facility      Init. Prod.                              10  Pending
  9 Init. Prod.   Test Market  Marketing     Evaluate      10  Pending
 10 Evaluate      Changes                                  10  Pending
 11 Test Market   Changes                                  15  Pending
 12 Changes       Production                                5  Pending
 13 Production                                              0  Pending
 14 Marketing                                               0  Pending

Obs A_DUR   A_START   A_FINISH   E_START   E_FINISH   L_START   L_FINISH

  1    5    02DEC91   06DEC91    02DEC91   06DEC91    02DEC91   06DEC91
  2   11    07DEC91   17DEC91    07DEC91   17DEC91    07DEC91   17DEC91
  3    5    06DEC91   10DEC91    06DEC91   10DEC91    06DEC91   10DEC91
  4    6    08DEC91   13DEC91    08DEC91   13DEC91    08DEC91   13DEC91
  5    .          .         .    20DEC91   07JAN92    20DEC91   07JAN92
  6    .    11DEC91         .    11DEC91   22DEC91    11DEC91   22DEC91
  7    .          .         .    08JAN92   17JAN92    08JAN92   17JAN92
  8    .          .         .    08JAN92   17JAN92    08JAN92   17JAN92
  9    .          .         .    18JAN92   27JAN92    18JAN92   27JAN92
 10    .          .         .    28JAN92   06FEB92    02FEB92   11FEB92
 11    .          .         .    28JAN92   11FEB92    28JAN92   11FEB92
 12    .          .         .    12FEB92   16FEB92    12FEB92   16FEB92
 13    .          .         .    17FEB92   17FEB92    17FEB92   17FEB92
 14    .          .         .    28JAN92   28JAN92    17FEB92   17FEB92

Obs T_FLOAT    F_FLOAT    B_START    B_FINISH    S_VAR    F_VAR

  1     0          0      02DEC91    06DEC91       0        0
  2     0          0      07DEC91    16DEC91       0        1
  3     0          0      07DEC91    11DEC91      -1       -1
  4     0          0      07DEC91    11DEC91       1        2
  5     0          0      17DEC91    04JAN92       3        3
  6     0          0      12DEC91    21DEC91      -1        1
  7     0          0      05JAN92    14JAN92       3        3
  8     0          0      05JAN92    14JAN92       3        3
  9     0          0      15JAN92    24JAN92       3        3
 10     5          5      25JAN92    03FEB92       3        3
 11     0          0      25JAN92    08FEB92       3        3
 12     0          0      09FEB92    13FEB92       3        3
 13     0          0      14FEB92    14FEB92       3        3
 14    20         20      25JAN92    25JAN92       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 2.13.4 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. Note that the activity 'Prototype' has no progress information in WIDGACT, but it is assumed to have an actual start date of December 18, 1991. 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, allows 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 'Anal. Market' has L_START='09JAN92' instead of '06DEC91', as in the earlier invocation of PROC CPM (without the SHOWFLOAT option).

Output 2.13.4. Comparison of Schedules: AUTOUPDT

```
                    Progress Update and Target Schedules
               Updated Schedule vs. Target Schedule: AUTOUPDT

Obs task          succ1        succ2           succ3     days  STATUS

  1 Approve Plan  Drawings     Anal. Market  Write Specs    5  Completed
  2 Drawings      Prototype                                10  Completed
  3 Anal. Market  Mkt. Strat.                               5  Completed
  4 Write Specs   Prototype                                 5  Completed
  5 Prototype     Materials    Facility                    15  In Progress
  6 Mkt. Strat.   Test Market  Marketing                   10  In Progress
  7 Materials     Init. Prod.                              10  Pending
  8 Facility      Init. Prod.                              10  Pending
  9 Init. Prod.   Test Market  Marketing     Evaluate      10  Pending
 10 Evaluate      Changes                                  10  Pending
 11 Test Market   Changes                                  15  Pending
 12 Changes       Production                                5  Pending
 13 Production                                              0  Pending
 14 Marketing                                               0  Pending

Obs A_DUR   A_START   A_FINISH   E_START   E_FINISH   L_START   L_FINISH

  1    5    02DEC91   06DEC91    02DEC91   06DEC91    02DEC91   06DEC91
  2   11    07DEC91   17DEC91    07DEC91   17DEC91    07DEC91   17DEC91
  3    5    06DEC91   10DEC91    06DEC91   10DEC91    09JAN92   13JAN92
  4    6    08DEC91   13DEC91    08DEC91   13DEC91    12DEC91   17DEC91
  5    .    18DEC91         .    18DEC91   05JAN92    18DEC91   05JAN92
  6    .    11DEC91         .    11DEC91   22DEC91    14JAN92   25JAN92
  7    .          .         .    06JAN92   15JAN92    06JAN92   15JAN92
  8    .          .         .    06JAN92   15JAN92    06JAN92   15JAN92
  9    .          .         .    16JAN92   25JAN92    16JAN92   25JAN92
 10    .          .         .    26JAN92   04FEB92    31JAN92   09FEB92
 11    .          .         .    26JAN92   09FEB92    26JAN92   09FEB92
 12    .          .         .    10FEB92   14FEB92    10FEB92   14FEB92
 13    .          .         .    15FEB92   15FEB92    15FEB92   15FEB92
 14    .          .         .    26JAN92   26JAN92    15FEB92   15FEB92

Obs T_FLOAT    F_FLOAT    B_START    B_FINISH    S_VAR    F_VAR

  1     0         -1      02DEC91    06DEC91        0        0
  2     0          0      07DEC91    16DEC91        0        1
  3    30          0      07DEC91    11DEC91       -1       -1
  4     4          4      07DEC91    11DEC91        1        2
  5     0          0      17DEC91    04JAN92        1        1
  6    30         30      12DEC91    21DEC91       -1        1
  7     0          0      05JAN92    14JAN92        1        1
  8     0          0      05JAN92    14JAN92        1        1
  9     0          0      15JAN92    24JAN92        1        1
 10     5          5      25JAN92    03FEB92        1        1
 11     0          0      25JAN92    08FEB92        1        1
 12     0          0      09FEB92    13FEB92        1        1
 13     0          0      14FEB92    14FEB92        1        1
 14    20         20      25JAN92    25JAN92        1        1
```

The following invocation of PROC CPM produced Output 2.13.4:

```
proc cpm data=widgact holidata=holidays
         out=widgupdt date='2dec91'd;
   activity task;
   succ     succ1 succ2 succ3;
   duration days;
   holiday  holiday / holifin=(holifin);
   baseline / compare=early;
   actual / as=sdate af=fdate timenow='20dec91'd
            remdur=rdur pctcomp=pctc
            autoupdt showfloat;
   run;
```

Example 2.14. 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 2.14.1.

Output 2.14.1. 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	Anal. 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 2.14.2. Resource Utilization: HOLDATA

Summarizing Resource Utilization
Holidays Data Set HOLDATA

Obs	hol	name
1	25DEC91	Christmas
2	01JAN92	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 via the Holiday data set, HOLDATA, which identifies two holidays, one for Christmas and one for New Year's Day. Output 2.14.2 shows the Holiday data set.

The program saves the resource usage information in a data set named ROUT, which is displayed in Output 2.14.3. 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='2dec91'd interval=weekday
         resourceout=rout data=widgres
         holidata=holdata;
   id task;
   tailnode tail;
   duration days;
   headnode head;
   resource engineer / maxdate='31dec91'd;
   holiday  hol;
   run;
```

Output 2.14.3. Resource Utilization: Resource Usage Data Set

Summarizing Resource Utilization
Resource Usage

Obs	_TIME_	Eengineer	Lengineer
1	02DEC91	2	2
2	03DEC91	2	2
3	04DEC91	2	2
4	05DEC91	2	2
5	06DEC91	2	2
6	09DEC91	4	1
7	10DEC91	4	1
8	11DEC91	4	1
9	12DEC91	4	1
10	13DEC91	4	1
11	16DEC91	1	3
12	17DEC91	1	3
13	18DEC91	1	3
14	19DEC91	1	3
15	20DEC91	1	3
16	23DEC91	4	4
17	24DEC91	4	4
18	26DEC91	4	4
19	27DEC91	4	4
20	30DEC91	4	4
21	31DEC91	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 2.14.4. Calendar Showing Resource Usage

```
                    Summarizing Resource Utilization
                             Resource Usage

        -------------------------------------------------------------
        |                                                           |
        |                     December  1991                        |
        |                                                           |
        |-----------------------------------------------------------|
        |  Monday   |  Tuesday  | Wednesday | Thursday  |  Friday   |
        |-----------+-----------+-----------+-----------+-----------|
        |     2     |     3     |     4     |     5     |     6     |
        |           |           |           |           |           |
        | 2 ESS Eng | 2 ESS Eng | 2 ESS Eng | 2 ESS Eng | 2 ESS Eng |
        | 2 LSS Eng | 2 LSS Eng | 2 LSS Eng | 2 LSS Eng | 2 LSS Eng |
        |-----------+-----------+-----------+-----------+-----------|
        |     9     |    10     |    11     |    12     |    13     |
        |           |           |           |           |           |
        | 4 ESS Eng | 4 ESS Eng | 4 ESS Eng | 4 ESS Eng | 4 ESS Eng |
        | 1 LSS Eng | 1 LSS Eng | 1 LSS Eng | 1 LSS Eng | 1 LSS Eng |
        |-----------+-----------+-----------+-----------+-----------|
        |    16     |    17     |    18     |    19     |    20     |
        |           |           |           |           |           |
        | 1 ESS Eng | 1 ESS Eng | 1 ESS Eng | 1 ESS Eng | 1 ESS Eng |
        | 3 LSS Eng | 3 LSS Eng | 3 LSS Eng | 3 LSS Eng | 3 LSS Eng |
        |-----------+-----------+-----------+-----------+-----------|
        |    23     |    24     |    25     |    26     |    27     |
        |           |           |*Christmas*|           |           |
        | 4 ESS Eng | 4 ESS Eng |           | 4 ESS Eng | 4 ESS Eng |
        | 4 LSS Eng | 4 LSS Eng |           | 4 LSS Eng | 4 LSS Eng |
        |-----------+-----------+-----------+-----------+-----------|
        |    30     |    31     |           |           |           |
        |           |           |           |           |           |
        | 4 ESS Eng | 4 ESS Eng |           |           |           |
        | 4 LSS Eng | 4 LSS Eng |           |           |           |
        -------------------------------------------------------------

                       --------------------------
                       |         Legend         |
                       |                        |
                       | ESS Usage of  engineer |
                       | LSS Usage of  engineer |
                       --------------------------
```

Output 2.14.5. Bar Chart for Early Start Usage

```
                  Summarizing Resource Utilization
                           Resource Usage

   Period Identifier                               ESS Usage of  en
                                                                Sum
            |
  02DEC91   |********************                          2.000000
  03DEC91   |********************                          2.000000
  04DEC91   |********************                          2.000000
  05DEC91   |********************                          2.000000
  06DEC91   |********************                          2.000000
  09DEC91   |****************************************      4.000000
  10DEC91   |****************************************      4.000000
  11DEC91   |****************************************      4.000000
  12DEC91   |****************************************      4.000000
  13DEC91   |****************************************      4.000000
  16DEC91   |**********                                    1.000000
  17DEC91   |**********                                    1.000000
  18DEC91   |**********                                    1.000000
  19DEC91   |**********                                    1.000000
  20DEC91   |**********                                    1.000000
  23DEC91   |****************************************      4.000000
  24DEC91   |****************************************      4.000000
  26DEC91   |****************************************      4.000000
  27DEC91   |****************************************      4.000000
  30DEC91   |****************************************      4.000000
  31DEC91   |****************************************      4.000000
            |
            ----------+---------+---------+---------+
                      1         2         3         4

                     ESS Usage of  engineer
```

Output 2.14.6. Bar Chart for Late Start Usage

```
                  Summarizing Resource Utilization
                           Resource Usage

   Period Identifier                               LSS Usage of  en
                                                                Sum
            |
  02DEC91   |********************                          2.000000
  03DEC91   |********************                          2.000000
  04DEC91   |********************                          2.000000
  05DEC91   |********************                          2.000000
  06DEC91   |********************                          2.000000
  09DEC91   |**********                                    1.000000
  10DEC91   |**********                                    1.000000
  11DEC91   |**********                                    1.000000
  12DEC91   |**********                                    1.000000
  13DEC91   |**********                                    1.000000
  16DEC91   |******************************                3.000000
  17DEC91   |******************************                3.000000
  18DEC91   |******************************                3.000000
  19DEC91   |******************************                3.000000
  20DEC91   |******************************                3.000000
  23DEC91   |****************************************      4.000000
  24DEC91   |****************************************      4.000000
  26DEC91   |****************************************      4.000000
  27DEC91   |****************************************      4.000000
  30DEC91   |****************************************      4.000000
  31DEC91   |****************************************      4.000000
            |
            ----------+---------+---------+---------+
                      1         2         3         4

                     LSS Usage of  engineer
```

Charts such as those shown in Output 2.14.4 through 2.14.6 can be used to compare different schedules with respect to resource usage.

Example 2.15. 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 2.14, 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 2.14 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 2.15.1.

Output 2.15.1. Resource Availability Data Set

Resource Allocation
Data Set WIDGRIN

Obs	per	otype	engineer	engcost
1	.	restype	1	2
2	.	suplevel	1	.
3	02DEC91	reslevel	3	40000
4	27DEC91	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 2, 1991. PROC CPM is then used to schedule the project to start on December 2, 1991, subject to the availability as specified.

```
proc cpm date='02dec91'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 2.15.2. Resource Constrained Schedule: Rule = SHORTDUR

Resource Allocation
Resource Constrained Schedule: Rule = SHORTDUR

Obs	tail	head	days	task	engineer	engcost	S_START	S_FINISH
1	1	2	5	Approve Plan	2	400	02DEC91	06DEC91
2	2	3	10	Drawings	1	200	16DEC91	30DEC91
3	2	4	5	Anal. Market	1	200	09DEC91	13DEC91
4	2	3	5	Write Specs	2	400	09DEC91	13DEC91
5	3	5	15	Prototype	4	800	31DEC91	21JAN92
6	4	6	10	Mkt. Strat.	.	.	16DEC91	30DEC91
7	5	7	10	Materials	.	.	22JAN92	04FEB92
8	5	7	10	Facility	2	400	22JAN92	04FEB92
9	7	8	10	Init. Prod.	4	800	05FEB92	18FEB92
10	8	9	10	Evaluate	1	200	19FEB92	03MAR92
11	6	9	15	Test Market	.	.	19FEB92	10MAR92
12	9	10	5	Changes	2	400	11MAR92	17MAR92
13	10	11	0	Production	4	800	18MAR92	18MAR92
14	6	12	0	Marketing	.	.	19FEB92	19FEB92
15	8	6	0	Dummy	.	.	19FEB92	19FEB92

Obs	E_START	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	02DEC91	06DEC91	02DEC91	06DEC91	0		
2	09DEC91	20DEC91	09DEC91	20DEC91	5	engineer	
3	09DEC91	13DEC91	22JAN92	28JAN92	0		
4	09DEC91	13DEC91	16DEC91	20DEC91	0		
5	23DEC91	14JAN92	23DEC91	14JAN92	0		
6	16DEC91	30DEC91	29JAN92	11FEB92	0		
7	15JAN92	28JAN92	15JAN92	28JAN92	0		
8	15JAN92	28JAN92	15JAN92	28JAN92	0		
9	29JAN92	11FEB92	29JAN92	11FEB92	0		
10	12FEB92	25FEB92	19FEB92	03MAR92	0		
11	12FEB92	03MAR92	12FEB92	03MAR92	0		
12	04MAR92	10MAR92	04MAR92	10MAR92	0		
13	11MAR92	11MAR92	11MAR92	11MAR92	0		
14	12FEB92	12FEB92	11MAR92	11MAR92	0		
15	12FEB92	12FEB92	12FEB92	12FEB92	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 2.15.2, contains the resource constrained start and finish times in the variables S_START and S_FINISH. On December 9, 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 16, 1991.

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.

Note that 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. Note that the entire project is delayed by five working days due to resource constraints (the maximum value of S_FINISH is 18MAR92, while the maximum value of E_FINISH is 11MAR92).

Note also that 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.

Output 2.15.3. Resource Usage: Rule = SHORTDUR

Resource Allocation
Usage Profiles for Constrained Schedule: Rule = SHORTDUR

Obs	_TIME_	Eengineer	Lengineer	Rengineer	Aengineer	Engcost	Lengcost	Rengcost	Aengcost
1	02DEC91	2	2	2	1	400	400	400	40000
2	03DEC91	2	2	2	1	400	400	400	39600
3	04DEC91	2	2	2	1	400	400	400	39200
4	05DEC91	2	2	2	1	400	400	400	38800
5	06DEC91	2	2	2	1	400	400	400	38400
6	09DEC91	4	1	3	0	800	200	600	38000
7	10DEC91	4	1	3	0	800	200	600	37400
8	11DEC91	4	1	3	0	800	200	600	36800
9	12DEC91	4	1	3	0	800	200	600	36200
10	13DEC91	4	1	3	0	800	200	600	35600
11	16DEC91	1	3	1	2	200	600	200	35000
12	17DEC91	1	3	1	2	200	600	200	34800
13	18DEC91	1	3	1	2	200	600	200	34600
14	19DEC91	1	3	1	2	200	600	200	34400
15	20DEC91	1	3	1	2	200	600	200	34200
16	23DEC91	4	4	1	2	800	800	200	34000
17	24DEC91	4	4	1	2	800	800	200	33800
18	26DEC91	4	4	1	2	800	800	200	33600
19	27DEC91	4	4	1	3	800	800	200	33400
20	30DEC91	4	4	1	3	800	800	200	33200
21	31DEC91	4	4	4	0	800	800	800	33000
22	02JAN92	4	4	4	0	800	800	800	32200
23	03JAN92	4	4	4	0	800	800	800	31400
24	06JAN92	4	4	4	0	800	800	800	30600
25	07JAN92	4	4	4	0	800	800	800	29800
26	08JAN92	4	4	4	0	800	800	800	29000
27	09JAN92	4	4	4	0	800	800	800	28200
28	10JAN92	4	4	4	0	800	800	800	27400
29	13JAN92	4	4	4	0	800	800	800	26600
30	14JAN92	4	4	4	0	800	800	800	25800
31	15JAN92	2	2	4	0	400	400	800	25000
32	16JAN92	2	2	4	0	400	400	800	24200
33	17JAN92	2	2	4	0	400	400	800	23400
34	20JAN92	2	2	4	0	400	400	800	22600
35	21JAN92	2	2	4	0	400	400	800	21800
36	22JAN92	2	3	2	2	400	600	400	21000
37	23JAN92	2	3	2	2	400	600	400	20600
38	24JAN92	2	3	2	2	400	600	400	20200
39	27JAN92	2	3	2	2	400	600	400	19800
40	28JAN92	2	3	2	2	400	600	400	19400
41	29JAN92	4	4	2	2	800	800	400	19000
42	30JAN92	4	4	2	2	800	800	400	18600
43	31JAN92	4	4	2	2	800	800	400	18200
44	03FEB92	4	4	2	2	800	800	400	17800
45	04FEB92	4	4	2	2	800	800	400	17400
46	05FEB92	4	4	4	0	800	800	800	17000
47	06FEB92	4	4	4	0	800	800	800	16200

Resource Allocation
Usage Profiles for Constrained Schedule: Rule = SHORTDUR

Obs	_TIME_	Eengineer	Lengineer	Rengineer	Aengineer	Eengcost	Lengcost	Rengcost	Aengcost
48	07FEB92	4	4	4	0	800	800	800	15400
49	10FEB92	4	4	4	0	800	800	800	14600
50	11FEB92	4	4	4	0	800	800	800	13800
51	12FEB92	1	0	4	0	200	0	800	13000
52	13FEB92	1	0	4	0	200	0	800	12200
53	14FEB92	1	0	4	0	200	0	800	11400
54	17FEB92	1	0	4	0	200	0	800	10600
55	18FEB92	1	0	4	0	200	0	800	9800
56	19FEB92	1	1	1	3	200	200	200	9000
57	20FEB92	1	1	1	3	200	200	200	8800
58	21FEB92	1	1	1	3	200	200	200	8600
59	24FEB92	1	1	1	3	200	200	200	8400
60	25FEB92	1	1	1	3	200	200	200	8200
61	26FEB92	0	1	1	3	0	200	200	8000
62	27FEB92	0	1	1	3	0	200	200	7800
63	28FEB92	0	1	1	3	0	200	200	7600
64	02MAR92	0	1	1	3	0	200	200	7400
65	03MAR92	0	1	1	3	0	200	200	7200
66	04MAR92	2	2	0	4	400	400	0	7000
67	05MAR92	2	2	0	4	400	400	0	7000
68	06MAR92	2	2	0	4	400	400	0	7000
69	09MAR92	2	2	0	4	400	400	0	7000
70	10MAR92	2	2	0	4	400	400	0	7000
71	11MAR92	0	0	2	2	0	0	400	7000
72	12MAR92	0	0	2	2	0	0	400	6600
73	13MAR92	0	0	2	2	0	0	400	6200
74	16MAR92	0	0	2	2	0	0	400	5800
75	17MAR92	0	0	2	2	0	0	400	5400
76	18MAR92	0	0	0	4	0	0	0	5000

The data set WIDGROUT, displayed in Output 2.15.3, 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 2.15.4. Resource Constrained Schedule: Rule = LST

```
                       Resource Allocation
           Resource Constrained Schedule: Rule = LST

Obs  tail  head  days  task          engineer  engcost  S_START  S_FINISH

  1    1     2     5   Approve Plan      2       400    02DEC91  06DEC91
  2    2     3    10   Drawings          1       200    09DEC91  20DEC91
  3    2     4     5   Anal. Market      1       200    16DEC91  20DEC91
  4    2     3     5   Write Specs       2       400    09DEC91  13DEC91
  5    3     5    15   Prototype         4       800    27DEC91  17JAN92
  6    4     6    10   Mkt. Strat.       .         .    23DEC91  07JAN92
  7    5     7    10   Materials         .         .    20JAN92  31JAN92
  8    5     7    10   Facility          2       400    20JAN92  31JAN92
  9    7     8    10   Init. Prod.       4       800    03FEB92  14FEB92
 10    8     9    10   Evaluate          1       200    17FEB92  28FEB92
 11    6     9    15   Test Market       .         .    17FEB92  06MAR92
 12    9    10     5   Changes           2       400    09MAR92  13MAR92
 13   10    11     0   Production        4       800    16MAR92  16MAR92
 14    6    12     0   Marketing         .         .    17FEB92  17FEB92
 15    8     6     0   Dummy             .         .    17FEB92  17FEB92

Obs  E_START   E_FINISH   L_START   L_FINISH   R_DELAY   DELAY_R    SUPPL_R

  1  02DEC91   06DEC91    02DEC91   06DEC91       0
  2  09DEC91   20DEC91    09DEC91   20DEC91       0
  3  09DEC91   13DEC91    22JAN92   28JAN92       5     engineer
  4  09DEC91   13DEC91    16DEC91   20DEC91       0
  5  23DEC91   14JAN92    23DEC91   14JAN92       3     engineer
  6  16DEC91   30DEC91    29JAN92   11FEB92       0
  7  15JAN92   28JAN92    15JAN92   28JAN92       0
  8  15JAN92   28JAN92    15JAN92   28JAN92       0
  9  29JAN92   11FEB92    29JAN92   11FEB92       0
 10  12FEB92   25FEB92    19FEB92   03MAR92       0
 11  12FEB92   03MAR92    12FEB92   03MAR92       0
 12  04MAR92   10MAR92    04MAR92   10MAR92       0
 13  11MAR92   11MAR92    11MAR92   11MAR92       0
 14  12FEB92   12FEB92    11MAR92   11MAR92       0
 15  12FEB92   12FEB92    12FEB92   12FEB92       0
```

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. Once again the variables DELAY_R and R_DELAY indicate that the resource engineer caused the activity 'Anal. Market' ('Prototype') to be delayed by five days (three days). However, the entire project is delayed only by three working days because the activity 'Anal. Market' is not a critical activity, and delaying it by five days did not affect the project completion time. The schedule and the resource usage data sets are displayed in Output 2.15.4 and Output 2.15.5, respectively.

```
proc cpm date='02dec91'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 2.15.5. Resource Usage: Rule = LST

Resource Allocation
Usage Profiles for Constrained Schedule: Rule = LST

Obs	_TIME_	Eengineer	Lengineer	Rengineer	Aengineer	Eengcost	Lengcost	Rengcost	Aengcost
1	02DEC91	2	2	2	1	400	400	400	40000
2	03DEC91	2	2	2	1	400	400	400	39600
3	04DEC91	2	2	2	1	400	400	400	39200
4	05DEC91	2	2	2	1	400	400	400	38800
5	06DEC91	2	2	2	1	400	400	400	38400
6	09DEC91	4	1	3	0	800	200	600	38000
7	10DEC91	4	1	3	0	800	200	600	37400
8	11DEC91	4	1	3	0	800	200	600	36800
9	12DEC91	4	1	3	0	800	200	600	36200
10	13DEC91	4	1	3	0	800	200	600	35600
11	16DEC91	1	3	2	1	200	600	400	35000
12	17DEC91	1	3	2	1	200	600	400	34600
13	18DEC91	1	3	2	1	200	600	400	34200
14	19DEC91	1	3	2	1	200	600	400	33800
15	20DEC91	1	3	2	1	200	600	400	33400
16	23DEC91	4	4	0	3	800	800	0	33000
17	24DEC91	4	4	0	3	800	800	0	33000
18	26DEC91	4	4	0	3	800	800	0	33000
19	27DEC91	4	4	4	0	800	800	800	33000
20	30DEC91	4	4	4	0	800	800	800	32200
21	31DEC91	4	4	4	0	800	800	800	31400
22	02JAN92	4	4	4	0	800	800	800	30600
23	03JAN92	4	4	4	0	800	800	800	29800
24	06JAN92	4	4	4	0	800	800	800	29000
25	07JAN92	4	4	4	0	800	800	800	28200
26	08JAN92	4	4	4	0	800	800	800	27400
27	09JAN92	4	4	4	0	800	800	800	26600
28	10JAN92	4	4	4	0	800	800	800	25800
29	13JAN92	4	4	4	0	800	800	800	25000
30	14JAN92	4	4	4	0	800	800	800	24200
31	15JAN92	2	2	4	0	400	400	800	23400
32	16JAN92	2	2	4	0	400	400	800	22600
33	17JAN92	2	2	4	0	400	400	800	21800
34	20JAN92	2	2	2	2	400	400	400	21000
35	21JAN92	2	2	2	2	400	400	400	20600
36	22JAN92	2	3	2	2	400	600	400	20200
37	23JAN92	2	3	2	2	400	600	400	19800
38	24JAN92	2	3	2	2	400	600	400	19400
39	27JAN92	2	3	2	2	400	600	400	19000
40	28JAN92	2	3	2	2	400	600	400	18600
41	29JAN92	4	4	2	2	800	800	400	18200
42	30JAN92	4	4	2	2	800	800	400	17800
43	31JAN92	4	4	2	2	800	800	400	17400
44	03FEB92	4	4	4	0	800	800	800	17000
45	04FEB92	4	4	4	0	800	800	800	16200
46	05FEB92	4	4	4	0	800	800	800	15400
47	06FEB92	4	4	4	0	800	800	800	14600

Resource Allocation
Usage Profiles for Constrained Schedule: Rule = LST

Obs	_TIME_	Engineer	Lengineer	Rengineer	Aengineer	Engcost	Lengcost	Rengcost	Aengcost
48	07FEB92	4	4	4	0	800	800	800	13800
49	10FEB92	4	4	4	0	800	800	800	13000
50	11FEB92	4	4	4	0	800	800	800	12200
51	12FEB92	1	0	4	0	200	0	800	11400
52	13FEB92	1	0	4	0	200	0	800	10600
53	14FEB92	1	0	4	0	200	0	800	9800
54	17FEB92	1	0	1	3	200	0	200	9000
55	18FEB92	1	0	1	3	200	0	200	8800
56	19FEB92	1	1	1	3	200	200	200	8600
57	20FEB92	1	1	1	3	200	200	200	8400
58	21FEB92	1	1	1	3	200	200	200	8200
59	24FEB92	1	1	1	3	200	200	200	8000
60	25FEB92	1	1	1	3	200	200	200	7800
61	26FEB92	0	1	1	3	0	200	200	7600
62	27FEB92	0	1	1	3	0	200	200	7400
63	28FEB92	0	1	1	3	0	200	200	7200
64	02MAR92	0	1	0	4	0	200	0	7000
65	03MAR92	0	1	0	4	0	200	0	7000
66	04MAR92	2	2	0	4	400	400	0	7000
67	05MAR92	2	2	0	4	400	400	0	7000
68	06MAR92	2	2	0	4	400	400	0	7000
69	09MAR92	2	2	2	2	400	400	400	7000
70	10MAR92	2	2	2	2	400	400	400	6600
71	11MAR92	0	0	2	2	0	0	400	6200
72	12MAR92	0	0	2	2	0	0	400	5800
73	13MAR92	0	0	2	2	0	0	400	5400
74	16MAR92	0	0	0	4	0	0	0	5000

Example 2.16. Using Supplementary Resources

In this example, the same project as in Example 2.15 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 2.16.1 and Output 2.16.2, respectively.

```
title 'Using Supplementary Resources';
proc cpm date='02dec91'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 23, 24, 25, and 26, to allow the project to be completed 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 23 by using a supplementary level of the resource engineer.

Note that the supplementary level is used only if the activity would otherwise get delayed beyond L_START + DELAY. Thus, the activity 'Anal. Market' is delayed by five days (S_START = '16DEC91') and scheduled later than its early start time (E_START = '09DEC91'), 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 '22JAN92' 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 2.16.1. Resource-Constrained Schedule: Supplementary Resource

Using Supplementary Resources
Resource Constrained Schedule

Obs	tail	head	days	task	engineer	engcost	S_START	S_FINISH
1	1	2	5	Approve Plan	2	400	02DEC91	06DEC91
2	2	3	10	Drawings	1	200	09DEC91	20DEC91
3	2	4	5	Anal. Market	1	200	16DEC91	20DEC91
4	2	3	5	Write Specs	2	400	09DEC91	13DEC91
5	3	5	15	Prototype	4	800	23DEC91	14JAN92
6	4	6	10	Mkt. Strat.	.	.	23DEC91	07JAN92
7	5	7	10	Materials	.	.	15JAN92	28JAN92
8	5	7	10	Facility	2	400	15JAN92	28JAN92
9	7	8	10	Init. Prod.	4	800	29JAN92	11FEB92
10	8	9	10	Evaluate	1	200	12FEB92	25FEB92
11	6	9	15	Test Market	.	.	12FEB92	03MAR92
12	9	10	5	Changes	2	400	04MAR92	10MAR92
13	10	11	0	Production	4	800	11MAR92	11MAR92
14	6	12	0	Marketing	.	.	12FEB92	12FEB92
15	8	6	0	Dummy	.	.	12FEB92	12FEB92

Obs	E_START	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	02DEC91	06DEC91	02DEC91	06DEC91	0		
2	09DEC91	20DEC91	09DEC91	20DEC91	0		
3	09DEC91	13DEC91	22JAN92	28JAN92	5	engineer	
4	09DEC91	13DEC91	16DEC91	20DEC91	0		
5	23DEC91	14JAN92	23DEC91	14JAN92	0		engineer
6	16DEC91	30DEC91	29JAN92	11FEB92	0		
7	15JAN92	28JAN92	15JAN92	28JAN92	0		
8	15JAN92	28JAN92	15JAN92	28JAN92	0		
9	29JAN92	11FEB92	29JAN92	11FEB92	0		
10	12FEB92	25FEB92	19FEB92	03MAR92	0		
11	12FEB92	03MAR92	12FEB92	03MAR92	0		
12	04MAR92	10MAR92	04MAR92	10MAR92	0		
13	11MAR92	11MAR92	11MAR92	11MAR92	0		
14	12FEB92	12FEB92	11MAR92	11MAR92	0		
15	12FEB92	12FEB92	12FEB92	12FEB92	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 2.15. 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 8th and 9th of December equals the value for the beginning of the 7th 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 9, 1991, requiring one and two engineers, respectively. The 'Write Specs' activity finishes on the 13th, freeing up two engineers, whereas 'Drawings' finishes only on the 20th of December. Thus, the data set WIDGRO16 has Rengineer equal to '3' from 9DEC91 to 13DEC91 and then equal to '1' on the 14th and 15th of December. Another engineer is required by the activity 'Anal. Market' from December 16, 1991; thus the total usage from 16DEC91 to 20DEC91 is 2.

Output 2.16.2. Resource Usage: Supplementary Resources

Using Supplementary Resources
Usage Profiles for Constrained Schedule

Obs	_TIME_	Eengineer	Lengineer	Rengineer	Aengineer	Encgost	Lncgost	Rncgost	Ancgost
1	02DEC91	2	2	2	1	0	0	0	40000
2	03DEC91	2	2	2	1	400	400	400	39600
3	04DEC91	2	2	2	1	800	800	800	39200
4	05DEC91	2	2	2	1	1200	1200	1200	38800
5	06DEC91	2	2	2	1	1600	1600	1600	38400
6	07DEC91	0	0	0	3	2000	2000	2000	38000
7	08DEC91	0	0	0	3	2000	2000	2000	38000
8	09DEC91	4	1	3	0	2000	2000	2000	38000
9	10DEC91	4	1	3	0	2800	2200	2600	37400
10	11DEC91	4	1	3	0	3600	2400	3200	36800
11	12DEC91	4	1	3	0	4400	2600	3800	36200
12	13DEC91	4	1	3	0	5200	2800	4400	35600
13	14DEC91	1	1	1	2	6000	3000	5000	35000
14	15DEC91	1	1	1	2	6000	3000	5000	35000
15	16DEC91	1	3	2	1	6000	3000	5000	35000
16	17DEC91	1	3	2	1	6200	3600	5400	34600
17	18DEC91	1	3	2	1	6400	4200	5800	34200
18	19DEC91	1	3	2	1	6600	4800	6200	33800
19	20DEC91	1	3	2	1	6800	5400	6600	33400
20	21DEC91	0	0	0	3	7000	6000	7000	33000
21	22DEC91	0	0	0	3	7000	6000	7000	33000
22	23DEC91	4	4	4	-1	7000	6000	7000	33000
23	24DEC91	4	4	4	-1	7800	6800	7800	32200
24	25DEC91	4	4	4	-1	8600	7600	8600	31400
25	26DEC91	4	4	4	-1	8600	7600	8600	31400
26	27DEC91	4	4	4	0	9400	8400	9400	30600
27	28DEC91	4	4	4	0	10200	9200	10200	29800
28	29DEC91	4	4	4	0	10200	9200	10200	29800
29	30DEC91	4	4	4	0	10200	9200	10200	29800
30	31DEC91	4	4	4	0	11000	10000	11000	29000
31	01JAN92	4	4	4	0	11800	10800	11800	28200
32	02JAN92	4	4	4	0	11800	10800	11800	28200
33	03JAN92	4	4	4	0	12600	11600	12600	27400
34	04JAN92	4	4	4	0	13400	12400	13400	26600
35	05JAN92	4	4	4	0	13400	12400	13400	26600
36	06JAN92	4	4	4	0	13400	12400	13400	26600
37	07JAN92	4	4	4	0	14200	13200	14200	25800
38	08JAN92	4	4	4	0	15000	14000	15000	25000
39	09JAN92	4	4	4	0	15800	14800	15800	24200
40	10JAN92	4	4	4	0	16600	15600	16600	23400
41	11JAN92	4	4	4	0	17400	16400	17400	22600
42	12JAN92	4	4	4	0	17400	16400	17400	22600
43	13JAN92	4	4	4	0	17400	16400	17400	22600
44	14JAN92	4	4	4	0	18200	17200	18200	21800
45	15JAN92	2	2	2	2	19000	18000	19000	21000
46	16JAN92	2	2	2	2	19400	18400	19400	20600
47	17JAN92	2	2	2	2	19800	18800	19800	20200

Using Supplementary Resources
Usage Profiles for Constrained Schedule

Obs	_TIME_	Eengineer	Lengineer	Rengineer	Aengineer	Eengcost	Lengcost	Rengcost	Aengcost
48	18JAN92	2	2	2	2	20200	19200	20200	19800
49	19JAN92	2	2	2	2	20200	19200	20200	19800
50	20JAN92	2	2	2	2	20200	19200	20200	19800
51	21JAN92	2	2	2	2	20600	19600	20600	19400
52	22JAN92	2	3	2	2	21000	20000	21000	19000
53	23JAN92	2	3	2	2	21400	20600	21400	18600
54	24JAN92	2	3	2	2	21800	21200	21800	18200
55	25JAN92	2	3	2	2	22200	21800	22200	17800
56	26JAN92	2	3	2	2	22200	21800	22200	17800
57	27JAN92	2	3	2	2	22200	21800	22200	17800
58	28JAN92	2	3	2	2	22600	22400	22600	17400
59	29JAN92	4	4	4	0	23000	23000	23000	17000
60	30JAN92	4	4	4	0	23800	23800	23800	16200
61	31JAN92	4	4	4	0	24600	24600	24600	15400
62	01FEB92	4	4	4	0	25400	25400	25400	14600
63	02FEB92	4	4	4	0	25400	25400	25400	14600
64	03FEB92	4	4	4	0	25400	25400	25400	14600
65	04FEB92	4	4	4	0	26200	26200	26200	13800
66	05FEB92	4	4	4	0	27000	27000	27000	13000
67	06FEB92	4	4	4	0	27800	27800	27800	12200
68	07FEB92	4	4	4	0	28600	28600	28600	11400
69	08FEB92	4	4	4	0	29400	29400	29400	10600
70	09FEB92	4	4	4	0	29400	29400	29400	10600
71	10FEB92	4	4	4	0	29400	29400	29400	10600
72	11FEB92	4	4	4	0	30200	30200	30200	9800
73	12FEB92	1	0	1	3	31000	31000	31000	9000
74	13FEB92	1	0	1	3	31200	31000	31200	8800
75	14FEB92	1	0	1	3	31400	31000	31400	8600
76	15FEB92	1	0	1	3	31600	31000	31600	8400
77	16FEB92	1	0	1	3	31600	31000	31600	8400
78	17FEB92	1	0	1	3	31600	31000	31600	8400
79	18FEB92	1	0	1	3	31800	31000	31800	8200
80	19FEB92	1	1	1	3	32000	31000	32000	8000
81	20FEB92	1	1	1	3	32200	31200	32200	7800
82	21FEB92	1	1	1	3	32400	31400	32400	7600
83	22FEB92	1	1	1	3	32600	31600	32600	7400
84	23FEB92	1	1	1	3	32600	31600	32600	7400
85	24FEB92	1	1	1	3	32600	31600	32600	7400
86	25FEB92	1	1	1	3	32800	31800	32800	7200
87	26FEB92	0	1	0	4	33000	32000	33000	7000
88	27FEB92	0	1	0	4	33000	32200	33000	7000
89	28FEB92	0	1	0	4	33000	32400	33000	7000
90	29FEB92	0	1	0	4	33000	32600	33000	7000
91	01MAR92	0	1	0	4	33000	32600	33000	7000
92	02MAR92	0	1	0	4	33000	32600	33000	7000
93	03MAR92	0	1	0	4	33000	32800	33000	7000
94	04MAR92	2	2	2	2	33000	33000	33000	7000

Using Supplementary Resources
Usage Profiles for Constrained Schedule

Obs	_TIME_	Eengineer	Lengineer	Rengineer	Aengineer	Eengcost	Lengcost	Rengcost	Aengcost
95	05MAR92	2	2	2	2	33400	33400	33400	6600
96	06MAR92	2	2	2	2	33800	33800	33800	6200
97	07MAR92	2	2	2	2	34200	34200	34200	5800
98	08MAR92	2	2	2	2	34200	34200	34200	5800
99	09MAR92	2	2	2	2	34200	34200	34200	5800
100	10MAR92	2	2	2	2	34600	34600	34600	5400
101	11MAR92	0	0	0	4	35000	35000	35000	5000

Example 2.17. Use of the INFEASDIAGNOSTIC Option

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 2.17.1. 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 2.17.2 displays the RESOURCEIN= data set, RESIN17.

Output 2.17.1. 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	Anal. 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 2.17.2. Resourcein Data Set RESIN17

Use of the INFEASDIAGNOSTIC Option
Data Set RESIN17

Obs	per	otype	deseng	mktan	prodeng	money
1	.	restype	1	1	1	2
2	02DEC91	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 2.17.3 displays the resulting Schedule data set WIDGO17S, which shows that the project is delayed by five days because of this resource. Note that 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='02dec91'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 2.17.3. 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
1	1	2	5	Approve Plan	1	02DEC91	06DEC91	02DEC91
2	2	3	10	Drawings	1	09DEC91	20DEC91	09DEC91
3	2	4	5	Anal. Market	.	09DEC91	13DEC91	09DEC91
4	2	3	5	Write Specs	1	23DEC91	30DEC91	09DEC91
5	3	5	15	Prototype	1	31DEC91	21JAN92	23DEC91
6	4	6	10	Mkt. Strat.	.	16DEC91	30DEC91	16DEC91
7	5	7	10	Materials	.	22JAN92	04FEB92	15JAN92
8	5	7	10	Facility	.	22JAN92	04FEB92	15JAN92
9	7	8	10	Init. Prod.	.	05FEB92	18FEB92	29JAN92
10	8	9	10	Evaluate	1	19FEB92	03MAR92	12FEB92
11	6	9	15	Test Market	.	19FEB92	10MAR92	12FEB92
12	9	10	5	Changes	1	11MAR92	17MAR92	04MAR92
13	10	11	0	Production	1	18MAR92	18MAR92	11MAR92
14	6	12	0	Marketing	.	19FEB92	19FEB92	12FEB92
15	8	6	0	Dummy	.	19FEB92	19FEB92	12FEB92

Obs	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	06DEC91	02DEC91	06DEC91	0		
2	20DEC91	09DEC91	20DEC91	0		
3	13DEC91	22JAN92	28JAN92	0		
4	13DEC91	16DEC91	20DEC91	10	deseng	
5	14JAN92	23DEC91	14JAN92	0		
6	30DEC91	29JAN92	11FEB92	0		
7	28JAN92	15JAN92	28JAN92	0		
8	28JAN92	15JAN92	28JAN92	0		
9	11FEB92	29JAN92	11FEB92	0		
10	25FEB92	19FEB92	03MAR92	0		
11	03MAR92	12FEB92	03MAR92	0		
12	10MAR92	04MAR92	10MAR92	0		
13	11MAR92	11MAR92	11MAR92	0		
14	12FEB92	11MAR92	11MAR92	0		
15	12FEB92	12FEB92	12FEB92	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 has missing values for the resources mktan and money. The INFEASDIAGNOSTIC option allows CPM to assume an infinite supplementary level for all the 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='02dec91'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 2.17.4 shows the new resource-constrained schedule. With a maximum delay of five days the procedure schedules the activity 'Anal. Market' on January 22, 1992, 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 2.17.5, 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 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 22nd and 28th of January, 1992. The negative values for Amktan indicate the days when a market analyst is needed for the project. Since money is a consumable resource with 0 availability 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.

Output 2.17.4. 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
1	1	2	5	Approve Plan	1	1	1	200	02DEC91	06DEC91
2	2	3	10	Drawings	1	1	.	100	09DEC91	20DEC91
3	2	4	5	Anal. Market	.	1	1	100	22JAN92	28JAN92
4	2	3	5	Write Specs	1	1	.	150	23DEC91	30DEC91
5	3	5	15	Prototype	1	1	.	300	31DEC91	21JAN92
6	4	6	10	Mkt. Strat.	.	.	1	150	29JAN92	11FEB92
7	5	7	10	Materials	.	.	.	300	22JAN92	04FEB92
8	5	7	10	Facility	.	1	.	500	22JAN92	04FEB92
9	7	8	10	Init. Prod.	.	.	.	250	05FEB92	18FEB92
10	8	9	10	Evaluate	1	.	.	150	19FEB92	03MAR92
11	6	9	15	Test Market	.	.	1	200	19FEB92	10MAR92
12	9	10	5	Changes	1	1	.	200	11MAR92	17MAR92
13	10	11	0	Production	1	1	.	600	18MAR92	18MAR92
14	6	12	0	Marketing	.	.	1	.	19FEB92	19FEB92
15	8	6	0	Dummy	.	.	.	.	19FEB92	19FEB92

Obs	E_START	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	02DEC91	06DEC91	02DEC91	06DEC91	0		mktan
2	09DEC91	20DEC91	09DEC91	20DEC91	0		money
3	09DEC91	13DEC91	22JAN92	28JAN92	30	prodeng	prodeng
4	09DEC91	13DEC91	16DEC91	20DEC91	10	deseng	money
5	23DEC91	14JAN92	23DEC91	14JAN92	0		money
6	16DEC91	30DEC91	29JAN92	11FEB92	0		mktan
7	15JAN92	28JAN92	15JAN92	28JAN92	0		money
8	15JAN92	28JAN92	15JAN92	28JAN92	0		money
9	29JAN92	11FEB92	29JAN92	11FEB92	0		money
10	12FEB92	25FEB92	19FEB92	03MAR92	0		money
11	12FEB92	03MAR92	12FEB92	03MAR92	0		mktan
12	04MAR92	10MAR92	04MAR92	10MAR92	0		money
13	11MAR92	11MAR92	11MAR92	11MAR92	0		
14	12FEB92	12FEB92	11MAR92	11MAR92	0		
15	12FEB92	12FEB92	12FEB92	12FEB92	0		

Output 2.17.5. Resource Usage: Multiple Resources

Use of the INFEASDIAGNOSTIC Option
Usage Profile: Multiple Resources

Obs	_TIME_	Rdeseng	Adeseng	Rprodeng	Aprodeng	Rmktan	Amktan	Rmoney	Amoney
1	02DEC91	1	0	1	0	1	-1	0	0
2	03DEC91	1	0	1	0	1	-1	200	-200
3	04DEC91	1	0	1	0	1	-1	400	-400
4	05DEC91	1	0	1	0	1	-1	600	-600
5	06DEC91	1	0	1	0	1	-1	800	-800
6	09DEC91	1	0	1	0	0	0	1000	-1000
7	10DEC91	1	0	1	0	0	0	1100	-1100
8	11DEC91	1	0	1	0	0	0	1200	-1200
9	12DEC91	1	0	1	0	0	0	1300	-1300
10	13DEC91	1	0	1	0	0	0	1400	-1400
11	16DEC91	1	0	1	0	0	0	1500	-1500
12	17DEC91	1	0	1	0	0	0	1600	-1600
13	18DEC91	1	0	1	0	0	0	1700	-1700
14	19DEC91	1	0	1	0	0	0	1800	-1800
15	20DEC91	1	0	1	0	0	0	1900	-1900
16	23DEC91	1	0	1	0	0	0	2000	-2000
17	24DEC91	1	0	1	0	0	0	2150	-2150
18	26DEC91	1	0	1	0	0	0	2300	-2300
19	27DEC91	1	0	1	0	0	0	2450	-2450
20	30DEC91	1	0	1	0	0	0	2600	-2600
21	31DEC91	1	0	1	0	0	0	2750	-2750
22	02JAN92	1	0	1	0	0	0	3050	-3050
23	03JAN92	1	0	1	0	0	0	3350	-3350
24	06JAN92	1	0	1	0	0	0	3650	-3650
25	07JAN92	1	0	1	0	0	0	3950	-3950
26	08JAN92	1	0	1	0	0	0	4250	-4250
27	09JAN92	1	0	1	0	0	0	4550	-4550
28	10JAN92	1	0	1	0	0	0	4850	-4850
29	13JAN92	1	0	1	0	0	0	5150	-5150
30	14JAN92	1	0	1	0	0	0	5450	-5450
31	15JAN92	1	0	1	0	0	0	5750	-5750
32	16JAN92	1	0	1	0	0	0	6050	-6050
33	17JAN92	1	0	1	0	0	0	6350	-6350
34	20JAN92	1	0	1	0	0	0	6650	-6650
35	21JAN92	1	0	1	0	0	0	6950	-6950
36	22JAN92	0	1	2	-1	1	-1	7250	-7250
37	23JAN92	0	1	2	-1	1	-1	8150	-8150
38	24JAN92	0	1	2	-1	1	-1	9050	-9050
39	27JAN92	0	1	2	-1	1	-1	9950	-9950
40	28JAN92	0	1	2	-1	1	-1	10850	-10850
41	29JAN92	0	1	1	0	1	-1	11750	-11750
42	30JAN92	0	1	1	0	1	-1	12700	-12700
43	31JAN92	0	1	1	0	1	-1	13650	-13650
44	03FEB92	0	1	1	0	1	-1	14600	-14600
45	04FEB92	0	1	1	0	1	-1	15550	-15550
46	05FEB92	0	1	0	1	1	-1	16500	-16500
47	06FEB92	0	1	0	1	1	-1	16900	-16900
48	07FEB92	0	1	0	1	1	-1	17300	-17300
49	10FEB92	0	1	0	1	1	-1	17700	-17700
50	11FEB92	0	1	0	1	1	-1	18100	-18100
51	12FEB92	0	1	0	1	0	0	18500	-18500
52	13FEB92	0	1	0	1	0	0	18750	-18750
53	14FEB92	0	1	0	1	0	0	19000	-19000
54	17FEB92	0	1	0	1	0	0	19250	-19250
55	18FEB92	0	1	0	1	0	0	19500	-19500

Use of the INFEASDIAGNOSTIC Option
Usage Profile: Multiple Resources

Obs	_TIME_	Rdeseng	Adeseng	Rprodeng	Aprodeng	Rmktan	Amktan	Rmoney	Amoney
56	19FEB92	1	0	0	1	1	-1	19750	-19750
57	20FEB92	1	0	0	1	1	-1	20100	-20100
58	21FEB92	1	0	0	1	1	-1	20450	-20450
59	24FEB92	1	0	0	1	1	-1	20800	-20800
60	25FEB92	1	0	0	1	1	-1	21150	-21150
61	26FEB92	1	0	0	1	1	-1	21500	-21500
62	27FEB92	1	0	0	1	1	-1	21850	-21850
63	28FEB92	1	0	0	1	1	-1	22200	-22200
64	02MAR92	1	0	0	1	1	-1	22550	-22550
65	03MAR92	1	0	0	1	1	-1	22900	-22900
66	04MAR92	0	1	0	1	1	-1	23250	-23250
67	05MAR92	0	1	0	1	1	-1	23450	-23450
68	06MAR92	0	1	0	1	1	-1	23650	-23650
69	09MAR92	0	1	0	1	1	-1	23850	-23850
70	10MAR92	0	1	0	1	1	-1	24050	-24050
71	11MAR92	1	0	1	0	0	0	24250	-24250
72	12MAR92	1	0	1	0	0	0	24450	-24450
73	13MAR92	1	0	1	0	0	0	24650	-24650
74	16MAR92	1	0	1	0	0	0	24850	-24850
75	17MAR92	1	0	1	0	0	0	25050	-25050
76	18MAR92	0	1	0	1	0	0	25250	-25250

Example 2.18. Variable Activity Delay

In Example 2.17, 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 2.17 with a slightly different scenario. Suppose that no delay is allowed in activities that require a production engineer. Data set WIDGR18, displayed in Output 2.18.1, 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 2.18.1. 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	Anal. 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 2.18.2 and Output 2.18.3.

```
data resin17;
   input per date7. otype $ 11-18
         deseng mktan prodeng money;
   format per date7.;
   datalines;
.         restype  1  1 1 2
02dec91   reslevel 1  . 1 .
;

data holdata;
   format hol date7.;
   input hol date7. name $ 10-18;
   datalines;
25dec91  Christmas
01jan92  New Year
;

proc cpm date='02dec91'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 2.18.2. Resource-Constrained Schedule: Variable Activity Delay

Variable Activity Delay
Resource Constrained Schedule

Obs	tail	head	days	task	adelay	deseng	prodeng	mktan	money	S_START
1	1	2	5	Approve Plan	0	1	1	1	200	02DEC91
2	2	3	10	Drawings	0	1	1	.	100	09DEC91
3	2	4	5	Anal. Market	0	.	1	1	100	15JAN92
4	2	3	5	Write Specs	0	1	1	.	150	09DEC91
5	3	5	15	Prototype	0	1	1	.	300	23DEC91
6	4	6	10	Mkt. Strat.	5	.	.	1	150	22JAN92
7	5	7	10	Materials	5	.	.	.	300	15JAN92
8	5	7	10	Facility	0	.	1	.	500	15JAN92
9	7	8	10	Init. Prod.	5	.	.	.	250	29JAN92
10	8	9	10	Evaluate	5	1	.	.	150	12FEB92
11	6	9	15	Test Market	5	.	.	1	200	12FEB92
12	9	10	5	Changes	0	1	1	.	200	04MAR92
13	10	11	0	Production	0	1	1	.	600	11MAR92
14	6	12	0	Marketing	5	.	.	1	.	12FEB92
15	8	6	0	Dummy	5	.	.	.	.	12FEB92

Obs	S_FINISH	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	R_DELAY	DELAY_R	SUPPL_R
1	06DEC91	02DEC91	06DEC91	02DEC91	06DEC91	0	0		mktan
2	20DEC91	09DEC91	20DEC91	09DEC91	20DEC91	0	0		money
3	21JAN92	09DEC91	13DEC91	22JAN92	28JAN92	30	25	prodeng	prodeng
4	13DEC91	09DEC91	13DEC91	16DEC91	20DEC91	5	0		deseng
5	14JAN92	23DEC91	14JAN92	23DEC91	14JAN92	0	0		money
6	04FEB92	16DEC91	30DEC91	29JAN92	11FEB92	30	0		mktan
7	28JAN92	15JAN92	28JAN92	15JAN92	28JAN92	0	0		money
8	28JAN92	15JAN92	28JAN92	15JAN92	28JAN92	0	0		money
9	11FEB92	29JAN92	11FEB92	29JAN92	11FEB92	0	0		money
10	25FEB92	12FEB92	25FEB92	19FEB92	03MAR92	5	0		money
11	03MAR92	12FEB92	03MAR92	12FEB92	03MAR92	0	0		mktan
12	10MAR92	04MAR92	10MAR92	04MAR92	10MAR92	0	0		money
13	11MAR92	11MAR92	11MAR92	11MAR92	11MAR92	0	0		
14	12FEB92	12FEB92	12FEB92	11MAR92	11MAR92	20	0		
15	12FEB92	12FEB92	12FEB92	12FEB92	12FEB92	0	0		

Output 2.18.3. Resource Usage

Variable Activity Delay
Usage Profile

Obs	_TIME_	Rdeseng	Adeseng	Rprodeng	Aprodeng	Rmktan	Amktan	Rmoney	Amoney
1	02DEC91	1	0	1	0	1	-1	0	0
2	03DEC91	1	0	1	0	1	-1	200	-200
3	04DEC91	1	0	1	0	1	-1	400	-400
4	05DEC91	1	0	1	0	1	-1	600	-600
5	06DEC91	1	0	1	0	1	-1	800	-800
6	09DEC91	2	-1	2	-1	0	0	1000	-1000
7	10DEC91	2	-1	2	-1	0	0	1250	-1250
8	11DEC91	2	-1	2	-1	0	0	1500	-1500
9	12DEC91	2	-1	2	-1	0	0	1750	-1750
10	13DEC91	2	-1	2	-1	0	0	2000	-2000
11	16DEC91	1	0	1	0	0	0	2250	-2250
12	17DEC91	1	0	1	0	0	0	2350	-2350
13	18DEC91	1	0	1	0	0	0	2450	-2450
14	19DEC91	1	0	1	0	0	0	2550	-2550
15	20DEC91	1	0	1	0	0	0	2650	-2650
16	23DEC91	1	0	1	0	0	0	2750	-2750
17	24DEC91	1	0	1	0	0	0	3050	-3050
18	26DEC91	1	0	1	0	0	0	3350	-3350
19	27DEC91	1	0	1	0	0	0	3650	-3650
20	30DEC91	1	0	1	0	0	0	3950	-3950
21	31DEC91	1	0	1	0	0	0	4250	-4250
22	02JAN92	1	0	1	0	0	0	4550	-4550
23	03JAN92	1	0	1	0	0	0	4850	-4850
24	06JAN92	1	0	1	0	0	0	5150	-5150
25	07JAN92	1	0	1	0	0	0	5450	-5450
26	08JAN92	1	0	1	0	0	0	5750	-5750
27	09JAN92	1	0	1	0	0	0	6050	-6050
28	10JAN92	1	0	1	0	0	0	6350	-6350
29	13JAN92	1	0	1	0	0	0	6650	-6650
30	14JAN92	1	0	1	0	0	0	6950	-6950
31	15JAN92	0	1	2	-1	1	-1	7250	-7250
32	16JAN92	0	1	2	-1	1	-1	8150	-8150
33	17JAN92	0	1	2	-1	1	-1	9050	-9050
34	20JAN92	0	1	2	-1	1	-1	9950	-9950
35	21JAN92	0	1	2	-1	1	-1	10850	-10850
36	22JAN92	0	1	1	0	1	-1	11750	-11750
37	23JAN92	0	1	1	0	1	-1	12700	-12700
38	24JAN92	0	1	1	0	1	-1	13650	-13650
39	27JAN92	0	1	1	0	1	-1	14600	-14600
40	28JAN92	0	1	1	0	1	-1	15550	-15550
41	29JAN92	0	1	0	1	1	-1	16500	-16500
42	30JAN92	0	1	0	1	1	-1	16900	-16900
43	31JAN92	0	1	0	1	1	-1	17300	-17300
44	03FEB92	0	1	0	1	1	-1	17700	-17700
45	04FEB92	0	1	0	1	1	-1	18100	-18100
46	05FEB92	0	1	0	1	0	0	18500	-18500
47	06FEB92	0	1	0	1	0	0	18750	-18750
48	07FEB92	0	1	0	1	0	0	19000	-19000
49	10FEB92	0	1	0	1	0	0	19250	-19250
50	11FEB92	0	1	0	1	0	0	19500	-19500
51	12FEB92	1	0	0	1	1	-1	19750	-19750
52	13FEB92	1	0	0	1	1	-1	20100	-20100
53	14FEB92	1	0	0	1	1	-1	20450	-20450
54	17FEB92	1	0	0	1	1	-1	20800	-20800
55	18FEB92	1	0	0	1	1	-1	21150	-21150

Variable Activity Delay
Usage Profile

Obs	_TIME_	Rdeseng	Adeseng	Rprodeng	Aprodeng	Rmktan	Amktan	Rmoney	Amoney
56	19FEB92	1	0	0	1	1	-1	21500	-21500
57	20FEB92	1	0	0	1	1	-1	21850	-21850
58	21FEB92	1	0	0	1	1	-1	22200	-22200
59	24FEB92	1	0	0	1	1	-1	22550	-22550
60	25FEB92	1	0	0	1	1	-1	22900	-22900
61	26FEB92	0	1	0	1	1	-1	23250	-23250
62	27FEB92	0	1	0	1	1	-1	23450	-23450
63	28FEB92	0	1	0	1	1	-1	23650	-23650
64	02MAR92	0	1	0	1	1	-1	23850	-23850
65	03MAR92	0	1	0	1	1	-1	24050	-24050
66	04MAR92	1	0	1	0	0	0	24250	-24250
67	05MAR92	1	0	1	0	0	0	24450	-24450
68	06MAR92	1	0	1	0	0	0	24650	-24650
69	09MAR92	1	0	1	0	0	0	24850	-24850
70	10MAR92	1	0	1	0	0	0	25050	-25050
71	11MAR92	0	1	0	1	0	0	25250	-25250

Note from the Schedule data set that the activity 'Anal. Market' is scheduled to start on January 15, 1992, even though (L_START + adelay)=22JAN92. 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 15JAN92. Note that 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 15th and 21st 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 9th of December. The RESOURCEOUT data set indicates that an extra design engineer and an extra production engineer are needed from the 9th to the 13th 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 2.18.4. 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. Note that 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. Note that 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='02dec91'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 2.18.4. Resource-Constrained Schedule: Activity Delay = - (T_FLOAT)

Variable Activity Delay
Resource Constrained Schedule
Activity Delay = - (T_FLOAT)

Obs	tail	head	days	task	actdel	deseng	prodeng	mktan	money	S_START	S_FINISH	E_START
1	1	2	5	Approve Plan	0	1	1	1	200	02DEC91	06DEC91	02DEC91
2	2	3	10	Drawings	0	1	1	.	100	09DEC91	20DEC91	09DEC91
3	2	4	5	Anal. Market	-30	.	1	1	100	09DEC91	13DEC91	09DEC91
4	2	3	5	Write Specs	-5	1	1	.	150	09DEC91	13DEC91	09DEC91
5	3	5	15	Prototype	0	1	1	.	300	23DEC91	14JAN92	23DEC91
6	4	6	10	Mkt. Strat.	-30	.	.	1	150	16DEC91	30DEC91	16DEC91
7	5	7	10	Materials	0	.	.	.	300	15JAN92	28JAN92	15JAN92
8	5	7	10	Facility	0	.	1	.	500	15JAN92	28JAN92	15JAN92
9	7	8	10	Init. Prod.	0	.	.	.	250	29JAN92	11FEB92	29JAN92
10	8	9	10	Evaluate	-5	1	.	.	150	12FEB92	25FEB92	12FEB92
11	6	9	15	Test Market	0	.	.	1	200	12FEB92	03MAR92	12FEB92
12	9	10	5	Changes	0	1	1	.	200	04MAR92	10MAR92	04MAR92
13	10	11	0	Production	0	1	1	.	600	11MAR92	11MAR92	11MAR92
14	6	12	0	Marketing	-20	.	.	1	.	12FEB92	12FEB92	12FEB92
15	8	6	0	Dummy	0	.	.	.	.	12FEB92	12FEB92	12FEB92

Obs	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R	B_START	B_FINISH	S_VAR	F_VAR
1	06DEC91	02DEC91	06DEC91	0		mktan	02DEC91	06DEC91	0	0
2	20DEC91	09DEC91	20DEC91	0		money	09DEC91	20DEC91	0	0
3	13DEC91	22JAN92	28JAN92	0		prodeng	09DEC91	13DEC91	0	0
4	13DEC91	16DEC91	20DEC91	0		deseng	09DEC91	13DEC91	0	0
5	14JAN92	23DEC91	14JAN92	0		money	23DEC91	14JAN92	0	0
6	30DEC91	29JAN92	11FEB92	0		mktan	16DEC91	30DEC91	0	0
7	28JAN92	15JAN92	28JAN92	0		money	15JAN92	28JAN92	0	0
8	28JAN92	15JAN92	28JAN92	0		money	15JAN92	28JAN92	0	0
9	11FEB92	29JAN92	11FEB92	0		money	29JAN92	11FEB92	0	0
10	25FEB92	19FEB92	03MAR92	0		money	12FEB92	25FEB92	0	0
11	03MAR92	12FEB92	03MAR92	0		mktan	12FEB92	03MAR92	0	0
12	10MAR92	04MAR92	10MAR92	0		money	04MAR92	10MAR92	0	0
13	11MAR92	11MAR92	11MAR92	0			11MAR92	11MAR92	0	0
14	12FEB92	11MAR92	11MAR92	0			12FEB92	12FEB92	0	0
15	12FEB92	12FEB92	12FEB92	0			12FEB92	12FEB92	0	0

Example 2.19. 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 2.19.1, 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 2.19.2, indicates that both resources are replenishable, there is one production manager available from December 2, and the hardware is unavailable on the 11th and 12th of December (perhaps it is scheduled for maintenance or has been reserved for some other project).

Output 2.19.1. 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	Anal. Market	1	.	.
3	Approve Plan	5	Write Specs	1	.	.
4	Drawings	10	Prototype	.	1	1
5	Anal. 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 2.19.2. Activity Splitting: Resource Availability Data Set

Activity Splitting
Resource Availability Data Set

Obs	per	otype	prodman	hrdware
1	.	restype	1	1
2	02DEC91	reslevel	1	1
3	11DEC91	reslevel	.	0
4	13DEC91	reslevel	.	1

The project is first scheduled without allowing any of the activities to be split. The Schedule data set SCHED, displayed in Output 2.19.3, indicates that the project has been delayed by one week (five working days, since maximum S_FINISH = '18MAR9'1 while maximum E_FINISH = '11MAR92'). Note that 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='02dec91'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 2.19.3. Project Schedule: Splitting Not Allowed

Activity Splitting
Project Schedule: Splitting not Allowed

Obs	task	succ	days	prodman	hrdware	S_START	S_FINISH
1	Approve Plan	Drawings	5	1	.	02DEC91	06DEC91
2	Drawings	Prototype	10	.	1	13DEC91	27DEC91
3	Anal. Market	Mkt. Strat.	5	.	.	09DEC91	13DEC91
4	Write Specs	Prototype	5	.	.	09DEC91	13DEC91
5	Prototype	Materials	15	1	.	31DEC91	21JAN92
6	Mkt. Strat.	Test Market	10	1	.	16DEC91	30DEC91
7	Materials	Init. Prod.	10	.	.	22JAN92	04FEB92
8	Facility	Init. Prod.	10	.	.	22JAN92	04FEB92
9	Init. Prod.	Test Market	10	1	.	05FEB92	18FEB92
10	Evaluate	Changes	10	1	.	19FEB92	03MAR92
11	Test Market	Changes	15	.	.	19FEB92	10MAR92
12	Changes	Production	5	.	.	11MAR92	17MAR92
13	Production		0	1	.	18MAR92	18MAR92
14	Marketing		0	.	.	19FEB92	19FEB92

Obs	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	02DEC91	06DEC91	02DEC91	06DEC91	0	0
2	09DEC91	20DEC91	09DEC91	20DEC91	0	0
3	09DEC91	13DEC91	22JAN92	28JAN92	30	0
4	09DEC91	13DEC91	16DEC91	20DEC91	5	5
5	23DEC91	14JAN92	23DEC91	14JAN92	0	0
6	16DEC91	30DEC91	29JAN92	11FEB92	30	30
7	15JAN92	28JAN92	15JAN92	28JAN92	0	0
8	15JAN92	28JAN92	15JAN92	28JAN92	0	0
9	29JAN92	11FEB92	29JAN92	11FEB92	0	0
10	12FEB92	25FEB92	19FEB92	03MAR92	5	5
11	12FEB92	03MAR92	12FEB92	03MAR92	0	0
12	04MAR92	10MAR92	04MAR92	10MAR92	0	0
13	11MAR92	11MAR92	11MAR92	11MAR92	0	0
14	12FEB92	12FEB92	11MAR92	11MAR92	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 allows the algorithm to split the 'Drawings' activity so that some of it is done before December 11, 1991, and the rest is scheduled to start on December 13, 1991. Likewise, the production manager is allocated to the activity 'Mkt. Strat.' on December 16, 1991. On the 26th 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 17th of January after the completion of 'Prototype' on the 16th of January. The Schedule data set, displayed in Output 2.19.4, contains separate observations for each segment of the split activities as indicated by the variable SEGMT_NO. Note that the project duration has been reduced by three working days, by allowing appropriate activities to be split.

```
proc cpm date='02dec91'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 2.19.4. 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	Anal. 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	02DEC91	06DEC91	02DEC91	06DEC91	02DEC91	06DEC91
2	09DEC91	24DEC91	09DEC91	20DEC91	09DEC91	20DEC91
3	09DEC91	10DEC91	09DEC91	20DEC91	09DEC91	20DEC91
4	13DEC91	24DEC91	09DEC91	20DEC91	09DEC91	20DEC91
5	09DEC91	13DEC91	09DEC91	13DEC91	22JAN92	28JAN92
6	09DEC91	13DEC91	09DEC91	13DEC91	16DEC91	20DEC91
7	26DEC91	16JAN92	23DEC91	14JAN92	23DEC91	14JAN92
8	16DEC91	21JAN92	16DEC91	30DEC91	29JAN92	11FEB92
9	16DEC91	24DEC91	16DEC91	30DEC91	29JAN92	11FEB92
10	17JAN92	21JAN92	16DEC91	30DEC91	29JAN92	11FEB92
11	17JAN92	30JAN92	15JAN92	28JAN92	15JAN92	28JAN92
12	17JAN92	30JAN92	15JAN92	28JAN92	15JAN92	28JAN92
13	31JAN92	13FEB92	29JAN92	11FEB92	29JAN92	11FEB92
14	14FEB92	27FEB92	12FEB92	25FEB92	19FEB92	03MAR92
15	14FEB92	05MAR92	12FEB92	03MAR92	12FEB92	03MAR92
16	06MAR92	12MAR92	04MAR92	10MAR92	04MAR92	10MAR92
17	13MAR92	13MAR92	11MAR92	11MAR92	11MAR92	11MAR92
18	14FEB92	14FEB92	12FEB92	12FEB92	11MAR92	11MAR92

Example 2.20. 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 2.20.1. As in Example 2.17, 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 2.20.1. 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	Anal. Market	1	1	.
3	Approve Plan	5	Write Specs	1	1	.
4	Drawings	10	Prototype	1	1	.
5	Anal. 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 2.20.2. 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	02DEC91	reslevel		1	1	4

The resource availability data set RESIN20, displayed in Output 2.20.2, 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 2.20.3. 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	.	02DEC91	06DEC91
2	Drawings	Prototype	10	1	1	.	09DEC91	20DEC91
3	Anal. Market	Mkt. Strat.	5	.	1	.	05FEB92	11FEB92
4	Write Specs	Prototype	5	1	1	.	23DEC91	30DEC91
5	Prototype	Materials	15	1	1	1	31DEC91	21JAN92
6	Mkt. Strat.	Test Market	10	.	.	.	12FEB92	25FEB92
7	Materials	Init. Prod.	10	.	.	.	22JAN92	04FEB92
8	Facility	Init. Prod.	10	.	1	2	22JAN92	04FEB92
9	Init. Prod.	Test Market	10	.	.	2	05FEB92	18FEB92
10	Evaluate	Changes	10	1	.	.	19FEB92	03MAR92
11	Test Market	Changes	15	.	.	.	26FEB92	17MAR92
12	Changes	Production	5	1	1	.	18MAR92	24MAR92
13	Production		0	.	.	.	25MAR92	25MAR92
14	Marketing		0	.	.	.	26FEB92	26FEB92

Obs	E_START	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	02DEC91	06DEC91	02DEC91	06DEC91	0		
2	09DEC91	20DEC91	09DEC91	20DEC91	0		
3	09DEC91	13DEC91	22JAN92	28JAN92	40	prodeng	
4	09DEC91	13DEC91	16DEC91	20DEC91	10	deseng	
5	23DEC91	14JAN92	23DEC91	14JAN92	0		
6	16DEC91	30DEC91	29JAN92	11FEB92	0		
7	15JAN92	28JAN92	15JAN92	28JAN92	0		
8	15JAN92	28JAN92	15JAN92	28JAN92	0		
9	29JAN92	11FEB92	29JAN92	11FEB92	0		
10	12FEB92	25FEB92	19FEB92	03MAR92	0		
11	12FEB92	03MAR92	12FEB92	03MAR92	0		
12	04MAR92	10MAR92	04MAR92	10MAR92	0		
13	11MAR92	11MAR92	11MAR92	11MAR92	0		
14	12FEB92	12FEB92	11MAR92	11MAR92	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 25, 1992 (see the schedule displayed in Output 2.20.3). The following program shows the invocation of PROC CPM.

```
proc cpm date='02dec91'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 2.20.4. The new schedule shows that the project completion time (11MAR92) has been reduced by two weeks as a result of using alternate resoruces.

Output 2.20.4. 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	.	02DEC91
2	Drawings	Prototype	10	1	1	.	1	1	.	09DEC91
3	Anal. Market	Mkt. Strat.	5	.	1	.	.	.	1	09DEC91
4	Write Specs	Prototype	5	1	1	.	.	.	2	09DEC91
5	Prototype	Materials	15	1	1	1	1	1	1	23DEC91
6	Mkt. Strat.	Test Market	10	.	.	.	.	.	.	16DEC91
7	Materials	Init. Prod.	10	.	.	.	.	.	.	15JAN92
8	Facility	Init. Prod.	10	.	1	2	.	1	2	15JAN92
9	Init. Prod.	Test Market	10	.	.	2	.	.	2	29JAN92
10	Evaluate	Changes	10	1	.	.	1	.	.	12FEB92
11	Test Market	Changes	15	.	.	.	.	.	.	12FEB92
12	Changes	Production	5	1	1	.	1	1	.	04MAR92
13	Production		0	.	.	.	.	.	.	11MAR92
14	Marketing		0	.	.	.	.	.	.	12FEB92

Obs	S_FINISH	E_START	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	06DEC91	02DEC91	06DEC91	02DEC91	06DEC91	0		
2	20DEC91	09DEC91	20DEC91	09DEC91	20DEC91	0		
3	13DEC91	09DEC91	13DEC91	22JAN92	28JAN92	0		
4	13DEC91	09DEC91	13DEC91	16DEC91	20DEC91	0		
5	14JAN92	23DEC91	14JAN92	23DEC91	14JAN92	0		
6	30DEC91	16DEC91	30DEC91	29JAN92	11FEB92	0		
7	28JAN92	15JAN92	28JAN92	15JAN92	28JAN92	0		
8	28JAN92	15JAN92	28JAN92	15JAN92	28JAN92	0		
9	11FEB92	29JAN92	11FEB92	29JAN92	11FEB92	0		
10	25FEB92	12FEB92	25FEB92	19FEB92	03MAR92	0		
11	03MAR92	12FEB92	03MAR92	12FEB92	03MAR92	0		
12	10MAR92	04MAR92	10MAR92	04MAR92	10MAR92	0		
13	11MAR92	11MAR92	11MAR92	11MAR92	11MAR92	0		
14	12FEB92	12FEB92	12FEB92	11MAR92	11MAR92	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*). Note that the activity 'Anal. Market' requires one production engineer who is tied up with the activity 'Drawings' on the 9th 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. The DELAYANALYSIS variables indicate that the supplementary levels are not used for any of the resources (recall that use of supplementary levels is triggered by the specification of a finite value for DELAY). It is evident from the project finish date (S_FINISH = L_FINISH = '11MAR92') that resource substitution has enabled the project to be completed without any delay.

The following program produced Output 2.20.4:

```
proc cpm date='02dec91'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 invocation of PROC CPM illustrates the use of both supplementary as well as alternate resources. Note from the output data set, displayed in Output 2.20.5, that once again the project is completed without any delay. Note also that the activity 'Write Specs' has used a supplementary resource whereas 'Anal. 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. The resulting schedule is displayed in Output 2.20.6.

```
proc cpm date='02dec91'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;
```

Output 2.20.5. 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	S_FINISH
1	Approve Plan	Drawings	5	1	1	.	1	1	.	02DEC91	06DEC91
2	Drawings	Prototype	10	1	1	.	1	1	.	09DEC91	20DEC91
3	Anal. Market	Mkt. Strat.	5	.	1	.	.	.	1	09DEC91	13DEC91
4	Write Specs	Prototype	5	1	1	.	1	1	.	09DEC91	13DEC91
5	Prototype	Materials	15	1	1	1	1	1	1	23DEC91	14JAN92
6	Mkt. Strat.	Test Market	10	.	.	.	.	.	.	16DEC91	30DEC91
7	Materials	Init. Prod.	10	.	.	.	.	.	.	15JAN92	28JAN92
8	Facility	Init. Prod.	10	.	1	2	.	1	2	15JAN92	28JAN92
9	Init. Prod.	Test Market	10	.	.	2	.	.	2	29JAN92	11FEB92
10	Evaluate	Changes	10	1	.	.	1	.	.	12FEB92	25FEB92
11	Test Market	Changes	15	.	.	.	.	.	.	12FEB92	03MAR92
12	Changes	Production	5	1	1	.	1	1	.	04MAR92	10MAR92
13	Production		0	.	.	.	.	.	.	11MAR92	11MAR92
14	Marketing		0	.	.	.	.	.	.	12FEB92	12FEB92

Obs	E_START	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	02DEC91	06DEC91	02DEC91	06DEC91	0		
2	09DEC91	20DEC91	09DEC91	20DEC91	0		
3	09DEC91	13DEC91	22JAN92	28JAN92	0		
4	09DEC91	13DEC91	16DEC91	20DEC91	0		deseng
5	23DEC91	14JAN92	23DEC91	14JAN92	0		
6	16DEC91	30DEC91	29JAN92	11FEB92	0		
7	15JAN92	28JAN92	15JAN92	28JAN92	0		
8	15JAN92	28JAN92	15JAN92	28JAN92	0		
9	29JAN92	11FEB92	29JAN92	11FEB92	0		
10	12FEB92	25FEB92	19FEB92	03MAR92	0		
11	12FEB92	03MAR92	12FEB92	03MAR92	0		
12	04MAR92	10MAR92	04MAR92	10MAR92	0		
13	11MAR92	11MAR92	11MAR92	11MAR92	0		
14	12FEB92	12FEB92	11MAR92	11MAR92	0		

Output 2.20.6. Alternate Resources Used before Supplementary Resources

Scheduling with Alternate Resources
DELAY=0, Alternate Resources Used instead of Supplementary

Obs	task	succ	days	desceng	prodeng	engpool	Udesceng	Uprodeng	Uengpool	S_START
1	Approve Plan	Drawings	5	1	1	.	1	1	.	02DEC91
2	Drawings	Prototype	10	1	1	.	1	1	.	09DEC91
3	Anal. Market	Mkt. Strat.	5	.	1	.	.	.	1	09DEC91
4	Write Specs	Prototype	5	1	1	.	.	.	2	09DEC91
5	Prototype	Materials	15	1	1	1	1	1	1	23DEC91
6	Mkt. Strat.	Test Market	10	.	.	.	.	.	.	16DEC91
7	Materials	Init. Prod.	10	.	.	.	.	.	.	15JAN92
8	Facility	Init. Prod.	10	.	1	2	.	1	2	15JAN92
9	Init. Prod.	Test Market	10	.	.	2	.	.	2	29JAN92
10	Evaluate	Changes	10	1	.	.	1	.	.	12FEB92
11	Test Market	Changes	15	.	.	.	.	.	.	12FEB92
12	Changes	Production	5	1	1	.	1	1	.	04MAR92
13	Production		0	.	.	.	.	.	.	11MAR92
14	Marketing		0	.	.	.	.	.	.	12FEB92

Obs	S_FINISH	E_START	E_FINISH	L_START	L_FINISH	R_DELAY	DELAY_R	SUPPL_R
1	06DEC91	02DEC91	06DEC91	02DEC91	06DEC91	0		
2	20DEC91	09DEC91	20DEC91	09DEC91	20DEC91	0		
3	13DEC91	09DEC91	13DEC91	22JAN92	28JAN92	0		
4	13DEC91	09DEC91	13DEC91	16DEC91	20DEC91	0		
5	14JAN92	23DEC91	14JAN92	23DEC91	14JAN92	0		
6	30DEC91	16DEC91	30DEC91	29JAN92	11FEB92	0		
7	28JAN92	15JAN92	28JAN92	15JAN92	28JAN92	0		
8	28JAN92	15JAN92	28JAN92	15JAN92	28JAN92	0		
9	11FEB92	29JAN92	11FEB92	29JAN92	11FEB92	0		
10	25FEB92	12FEB92	25FEB92	19FEB92	03MAR92	0		
11	03MAR92	12FEB92	03MAR92	12FEB92	03MAR92	0		
12	10MAR92	04MAR92	10MAR92	04MAR92	10MAR92	0		
13	11MAR92	11MAR92	11MAR92	11MAR92	11MAR92	0		
14	12FEB92	12FEB92	12FEB92	11MAR92	11MAR92	0		

Example 2.21. 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, apriori, 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. Note that 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
```

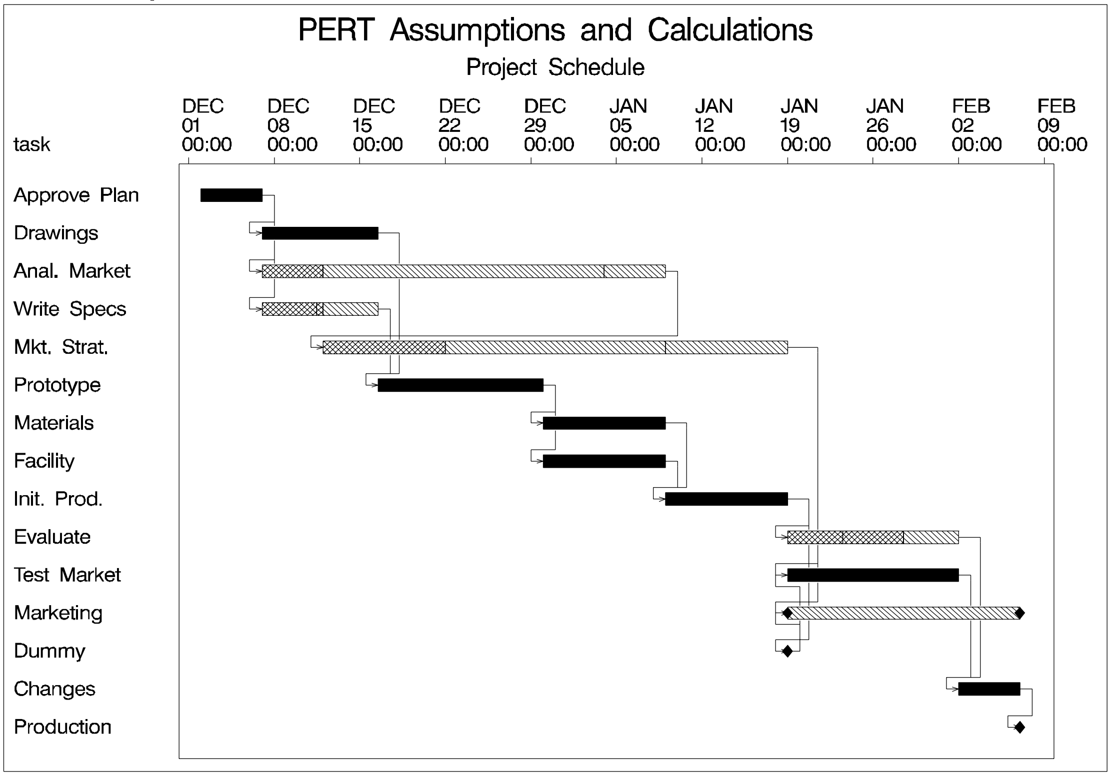
The following program saves the network (AOA format) from Example 2.2 with three estimates of activity durations in a SAS data set. The DATA step also calculates the weighted average duration for each activity. Following the DATA step, PROC CPM is invoked to produce the schedule plotted on a Gantt chart in Output 2.21.1. 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;
   input task $ 1-12 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
Anal. 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='2dec91'd;
   tailnode tail;
   headnode head;
   duration dur;
   id task;
   run;

proc sort;
   by e_start;
   run;

goptions vpos=50 hpos=80 border;
proc gantt graphics data=sched;
   chart / compress tailnode=tail headnode=head
           font=swiss height=1.5 nojobnum skip=2
           dur=dur increment=7 nolegend
           cframe=ligr;
   id task;
   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 2.21.1. 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 2.22. 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 Soft-*

ware: 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 2.22.1, 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 2.22.1. 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 2.22.2. The schedule can also be displayed graphically using the NETDRAW procedure, as illustrated in a similar problem in Example 5.16 in Chapter 5, "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 2.22.2. 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 2.23. 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 Figure 2.8.

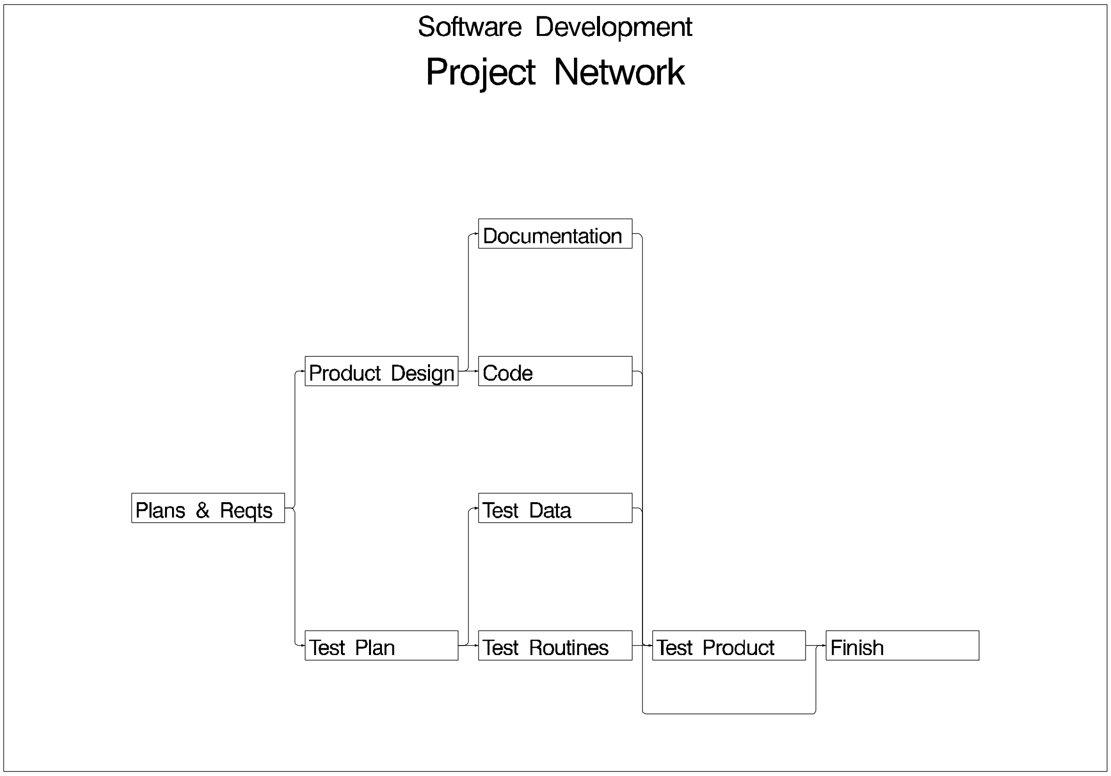


Figure 2.8. Software Project Network

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 2.23.1.

Output 2.23.1. Project Data

Software Development
Activity Data Set SOFTWARE

Activity	act	s1	s2	dur	mandays	progrmr	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 11, 1994, and the activities are assumed to follow a five-day work week. Unlike fixed-duration scheduling, each resource used by an activity could have a different schedule; an activity is assumed to be finished only when all of its resources have finished working on it.

```
proc cpm data=software out=sftout ressched=rsftout
         date='11apr94'd interval=weekday resout=rout;
   act act;
   succ s1 s2;
   dur dur;
   res progrmr 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 2.23.2) 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 2.23.2. Resource Schedule Data Set

Software Development
Resource Schedule Data Set RSFTOUT

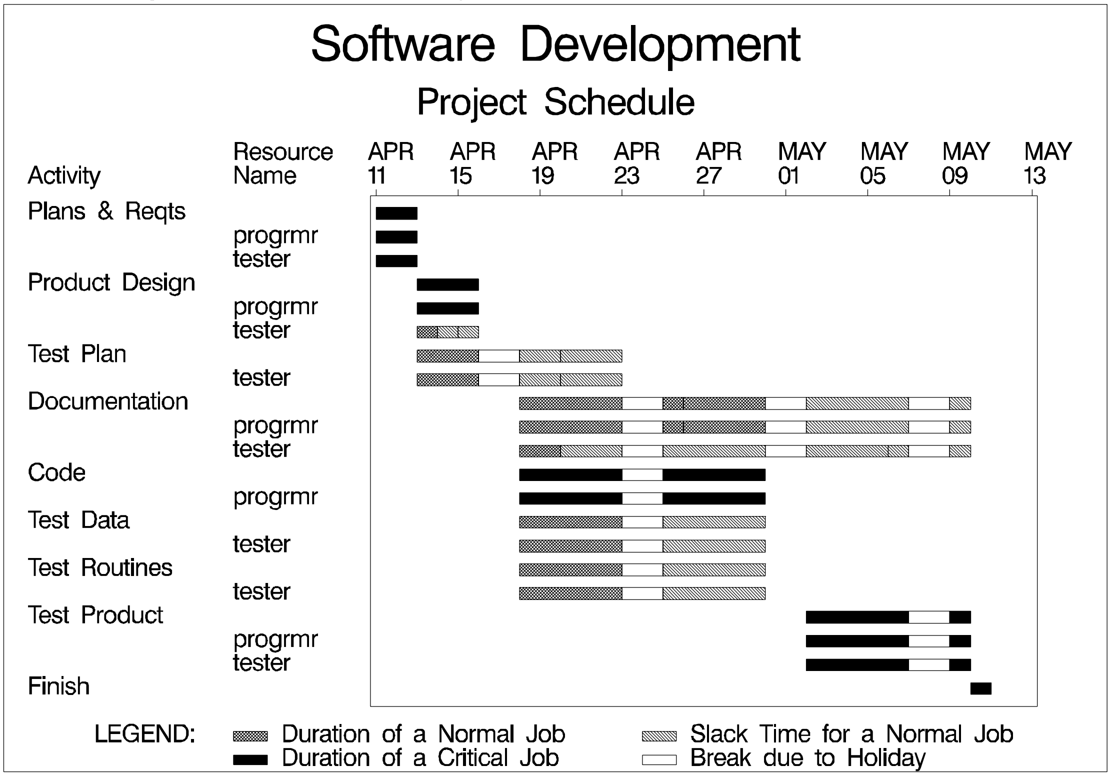
Activity	act	RESOURCE	DUR_TYPE	dur	mandays	R_RATE	E_START	E_FINISH	L_START	L_FINISH
Plans & Reqts	1			2	.	.	11APR94	12APR94	11APR94	12APR94
Plans & Reqts	1	progrmr	FIXED	2	.	1.0	11APR94	12APR94	11APR94	12APR94
Plans & Reqts	1	tester	FIXED	2	.	1.0	11APR94	12APR94	11APR94	12APR94
Product Design	2			3	.	.	13APR94	15APR94	13APR94	15APR94
Product Design	2	progrmr	RDRIVEN	3	3	1.0	13APR94	15APR94	13APR94	15APR94
Product Design	2	tester	RDRIVEN	1	1	1.0	13APR94	13APR94	15APR94	15APR94
Test Plan	3			3	.	.	13APR94	15APR94	20APR94	22APR94
Test Plan	3	tester	FIXED	3	.	1.0	13APR94	15APR94	20APR94	22APR94
Documentation	4			10	.	.	18APR94	29APR94	26APR94	09MAY94
Documentation	4	progrmr	RDRIVEN	10	2	0.2	18APR94	29APR94	26APR94	09MAY94
Documentation	4	tester	RDRIVEN	2	1	0.5	18APR94	19APR94	06MAY94	09MAY94
Code	5			10	.	.	18APR94	29APR94	18APR94	29APR94
Code	5	progrmr	FIXED	10	.	0.8	18APR94	29APR94	18APR94	29APR94
Test Data	6			5	.	.	18APR94	22APR94	25APR94	29APR94
Test Data	6	tester	FIXED	5	.	0.5	18APR94	22APR94	25APR94	29APR94
Test Routines	7			5	.	.	18APR94	22APR94	25APR94	29APR94
Test Routines	7	tester	FIXED	5	.	0.5	18APR94	22APR94	25APR94	29APR94
Test Product	8			6	.	.	02MAY94	09MAY94	02MAY94	09MAY94
Test Product	8	progrmr	FIXED	6	.	0.5	02MAY94	09MAY94	02MAY94	09MAY94
Test Product	8	tester	FIXED	6	.	1.0	02MAY94	09MAY94	02MAY94	09MAY94
Finish	9			0	.	.	10MAY94	10MAY94	10MAY94	10MAY94

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. Note that for driving resources, the variable dur is computed as (mandays / R_RATE).

A Gantt chart of the schedules for each resource is plotted in Output 2.23.3.

Output 2.23.3. Software Project Schedule



The daily utilization of the resources is also saved in a data set, ROUT, displayed in Output 2.23.4. The resource usage data set indicates that you need more than one tester on some days with both the early schedule (on the 13th, 18th, and 19th of April) and the late schedule (on the 6th and 9th of May).

Output 2.23.4. Resource Usage Data

Software Development
Resource Usage Data Set ROUT

Obs	_TIME_	Eprogrmr	Lprogrmr	Etester	Ltester
1	11APR94	1.0	1.0	1.0	1.0
2	12APR94	1.0	1.0	1.0	1.0
3	13APR94	1.0	1.0	2.0	0.0
4	14APR94	1.0	1.0	1.0	0.0
5	15APR94	1.0	1.0	1.0	1.0
6	18APR94	1.0	0.8	1.5	0.0
7	19APR94	1.0	0.8	1.5	0.0
8	20APR94	1.0	0.8	1.0	1.0
9	21APR94	1.0	0.8	1.0	1.0
10	22APR94	1.0	0.8	1.0	1.0
11	25APR94	1.0	0.8	0.0	1.0
12	26APR94	1.0	1.0	0.0	1.0
13	27APR94	1.0	1.0	0.0	1.0
14	28APR94	1.0	1.0	0.0	1.0
15	29APR94	1.0	1.0	0.0	1.0
16	02MAY94	0.5	0.7	1.0	1.0
17	03MAY94	0.5	0.7	1.0	1.0
18	04MAY94	0.5	0.7	1.0	1.0
19	05MAY94	0.5	0.7	1.0	1.0
20	06MAY94	0.5	0.7	1.0	1.5
21	09MAY94	0.5	0.7	1.0	1.5
22	10MAY94	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 2.23.5).

Output 2.23.5. Resource Availability Data

Software Development
Resource Availability Data Set

Obs	per	otype	progrmr	tester
1	11APR94	reslevel	1	1

The following statements invoke PROC CPM, and the resulting Resource Schedule data set is displayed in Output 2.23.6. Note that the project still finishes on May 10, but some of the activities (3, 4, 6, and 7) are delayed. The resource-constrained schedule is plotted on a Gantt chart in Output 2.23.7; both resources follow the same weekday calendar.

```
proc cpm data=software resin=resin
         out=sftout1 resout=rout1
         rsched=rsftout1
         date='11apr94'd interval=weekday;
   act act;
```

```
   succ s1 s2;
   dur dur;
   res progrmr tester / work=mandays
                        obstype=otype
                        period=per
                        rschedid=Activity;
   id Activity;
run;
```

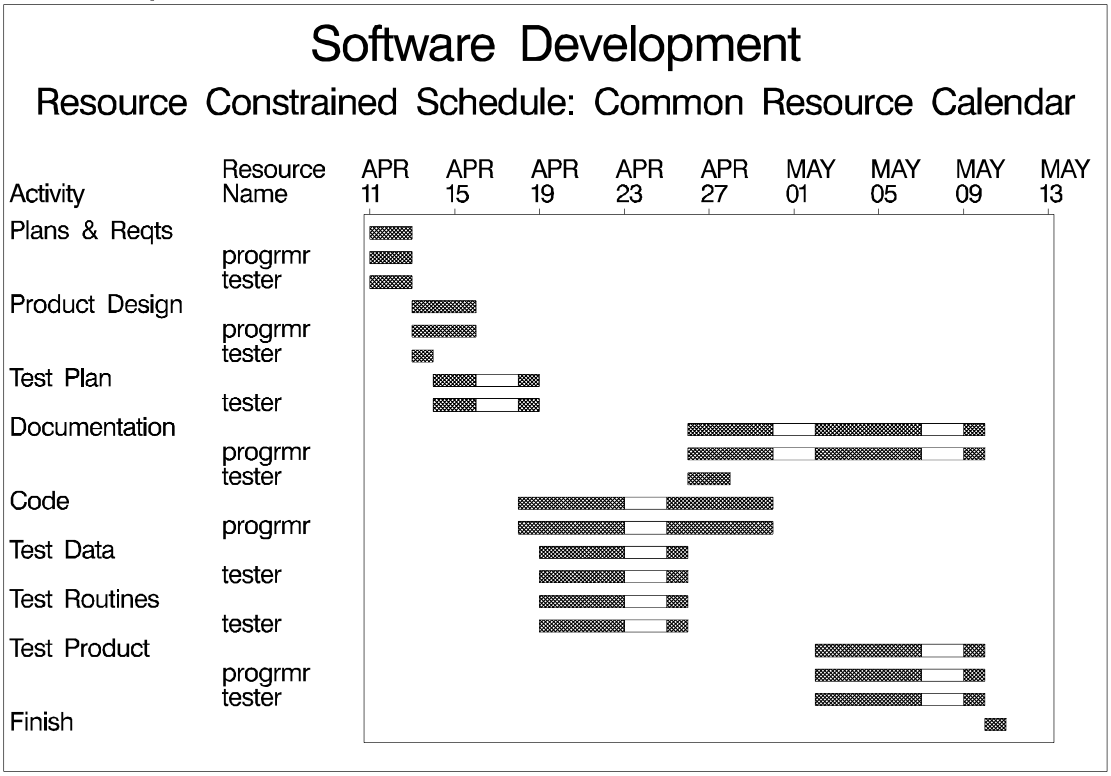
Output 2.23.6. Resource-Constrained Schedule: Common Calendar

Software Development
Resource Constrained Schedule: Common Resource Calendar

Activity	act	_CAL_	RESOURCE	DUR_TYPE	dur	mandays	R_RATE	S_START
Plans & Reqts	1	0			2	.	.	11APR94
Plans & Reqts	1	0	progrmr	FIXED	2	.	1.0	11APR94
Plans & Reqts	1	0	tester	FIXED	2	.	1.0	11APR94
Product Design	2	0			3	.	.	13APR94
Product Design	2	0	progrmr	RDRIVEN	3	3	1.0	13APR94
Product Design	2	0	tester	RDRIVEN	1	1	1.0	13APR94
Test Plan	3	0			3	.	.	14APR94
Test Plan	3	0	tester	FIXED	3	.	1.0	14APR94
Documentation	4	0			10	.	.	26APR94
Documentation	4	0	progrmr	RDRIVEN	10	2	0.2	26APR94
Documentation	4	0	tester	RDRIVEN	2	1	0.5	26APR94
Code	5	0			10	.	.	18APR94
Code	5	0	progrmr	FIXED	10	.	0.8	18APR94
Test Data	6	0			5	.	.	19APR94
Test Data	6	0	tester	FIXED	5	.	0.5	19APR94
Test Routines	7	0			5	.	.	19APR94
Test Routines	7	0	tester	FIXED	5	.	0.5	19APR94
Test Product	8	0			6	.	.	02MAY94
Test Product	8	0	progrmr	FIXED	6	.	0.5	02MAY94
Test Product	8	0	tester	FIXED	6	.	1.0	02MAY94
Finish	9	0			0	.	.	10MAY94

Activity	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
Plans & Reqts	12APR94	11APR94	12APR94	11APR94	12APR94
Plans & Reqts	12APR94	11APR94	12APR94	11APR94	12APR94
Plans & Reqts	12APR94	11APR94	12APR94	11APR94	12APR94
Product Design	15APR94	13APR94	15APR94	13APR94	15APR94
Product Design	15APR94	13APR94	15APR94	13APR94	15APR94
Product Design	13APR94	13APR94	13APR94	15APR94	15APR94
Test Plan	18APR94	13APR94	15APR94	20APR94	22APR94
Test Plan	18APR94	13APR94	15APR94	20APR94	22APR94
Documentation	09MAY94	18APR94	29APR94	26APR94	09MAY94
Documentation	09MAY94	18APR94	29APR94	26APR94	09MAY94
Documentation	27APR94	18APR94	19APR94	06MAY94	09MAY94
Code	29APR94	18APR94	29APR94	18APR94	29APR94
Code	29APR94	18APR94	29APR94	18APR94	29APR94
Test Data	25APR94	18APR94	22APR94	25APR94	29APR94
Test Data	25APR94	18APR94	22APR94	25APR94	29APR94
Test Routines	25APR94	18APR94	22APR94	25APR94	29APR94
Test Routines	25APR94	18APR94	22APR94	25APR94	29APR94
Test Product	09MAY94	02MAY94	09MAY94	02MAY94	09MAY94
Test Product	09MAY94	02MAY94	09MAY94	02MAY94	09MAY94
Test Product	09MAY94	02MAY94	09MAY94	02MAY94	09MAY94
Finish	10MAY94	10MAY94	10MAY94	10MAY94	10MAY94

Output 2.23.7. 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 2.23.8). In a new resource availability data set (also displayed in Output 2.23.8), associate calendar '1' with the resource tester and calendar '0' with the resource progrmr. Note that '0' refers to the default calendar, which is the weekday calendar for this project (since INTERVAL = WEEKDAY).

Output 2.23.8. Resource and Calendar Data

Software Development
Calendar Data Set CALENDAR

Obs	_cal_	_fri_
1	1	holiday

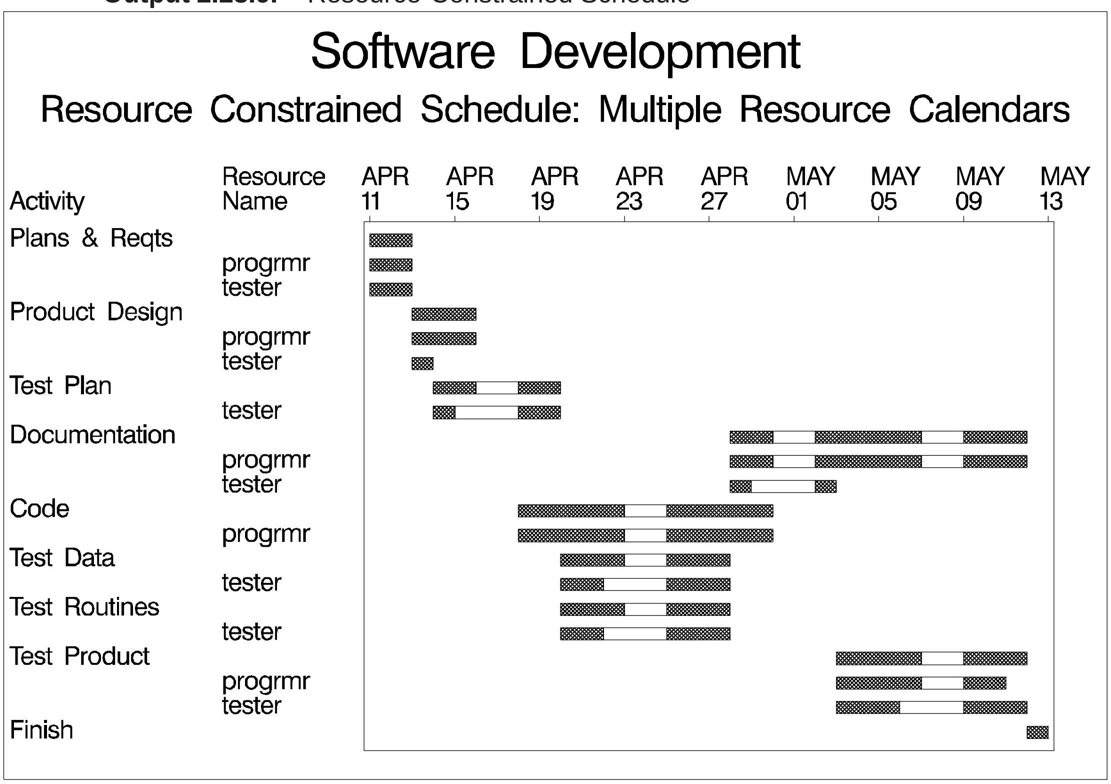
Resource Data Set RESIN2

Obs	per	otype	progrmr	tester
1	.	calendar	0	1
2	11APR94	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 2.23.9. Note that 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 2.23.10.

```
proc cpm data=software resin=resin2
         caledata=calendar
         out=sftout2 rsched=rsftout2
         resout=rout2
         date='11apr94'd interval=weekday;
   act act;
   succ s1 s2;
   dur dur;
   res progrmr tester / work=mandays
                        obstype=otype
                        period=per
                        rschedid=Activity;
   id Activity;
run;
```

Output 2.23.9. Resource-Constrained Schedule



Output 2.23.10. 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
Plans & Reqts	1	0			2	.	.	11APR94
Plans & Reqts	1	0	progrmr	FIXED	2	.	1.0	11APR94
Plans & Reqts	1	1	tester	FIXED	2	.	1.0	11APR94
Product Design	2	0			3	.	.	13APR94
Product Design	2	0	progrmr	RDRIVEN	3	3	1.0	13APR94
Product Design	2	1	tester	RDRIVEN	1	1	1.0	13APR94
Test Plan	3	0			3	.	.	14APR94
Test Plan	3	1	tester	FIXED	3	.	1.0	14APR94
Documentation	4	0			10	.	.	28APR94
Documentation	4	0	progrmr	RDRIVEN	10	2	0.2	28APR94
Documentation	4	1	tester	RDRIVEN	2	1	0.5	28APR94
Code	5	0			10	.	.	18APR94
Code	5	0	progrmr	FIXED	10	.	0.8	18APR94
Test Data	6	0			5	.	.	20APR94
Test Data	6	1	tester	FIXED	5	.	0.5	20APR94
Test Routines	7	0			5	.	.	20APR94
Test Routines	7	1	tester	FIXED	5	.	0.5	20APR94
Test Product	8	0			6	.	.	03MAY94
Test Product	8	0	progrmr	FIXED	6	.	0.5	03MAY94
Test Product	8	1	tester	FIXED	6	.	1.0	03MAY94
Finish	9	0			0	.	.	12MAY94

Activity	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
Plans & Reqts	12APR94	11APR94	12APR94	11APR94	12APR94
Plans & Reqts	12APR94	11APR94	12APR94	11APR94	12APR94
Plans & Reqts	12APR94	11APR94	12APR94	11APR94	12APR94
Product Design	15APR94	13APR94	15APR94	13APR94	15APR94
Product Design	15APR94	13APR94	15APR94	13APR94	15APR94
Product Design	13APR94	13APR94	13APR94	14APR94	14APR94
Test Plan	19APR94	13APR94	18APR94	18APR94	20APR94
Test Plan	19APR94	13APR94	18APR94	18APR94	20APR94
Documentation	11MAY94	18APR94	29APR94	27APR94	10MAY94
Documentation	11MAY94	18APR94	29APR94	27APR94	10MAY94
Documentation	02MAY94	18APR94	19APR94	09MAY94	10MAY94
Code	29APR94	18APR94	29APR94	18APR94	29APR94
Code	29APR94	18APR94	29APR94	18APR94	29APR94
Test Data	27APR94	19APR94	26APR94	21APR94	29APR94
Test Data	27APR94	19APR94	26APR94	21APR94	28APR94
Test Routines	27APR94	19APR94	26APR94	21APR94	29APR94
Test Routines	27APR94	19APR94	26APR94	21APR94	28APR94
Test Product	11MAY94	02MAY94	10MAY94	02MAY94	10MAY94
Test Product	10MAY94	02MAY94	09MAY94	03MAY94	10MAY94
Test Product	11MAY94	02MAY94	10MAY94	02MAY94	10MAY94
Finish	12MAY94	11MAY94	11MAY94	11MAY94	11MAY94

Example 2.24. 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 2.24.1.

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 2.24.1.

Output 2.24.1. 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     14APR95
```

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 14, 1995. 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 2.24.2, after being sorted by the variable ES_ASC.

Note that 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='3apr95'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 2.24.2. Survey Project Schedule

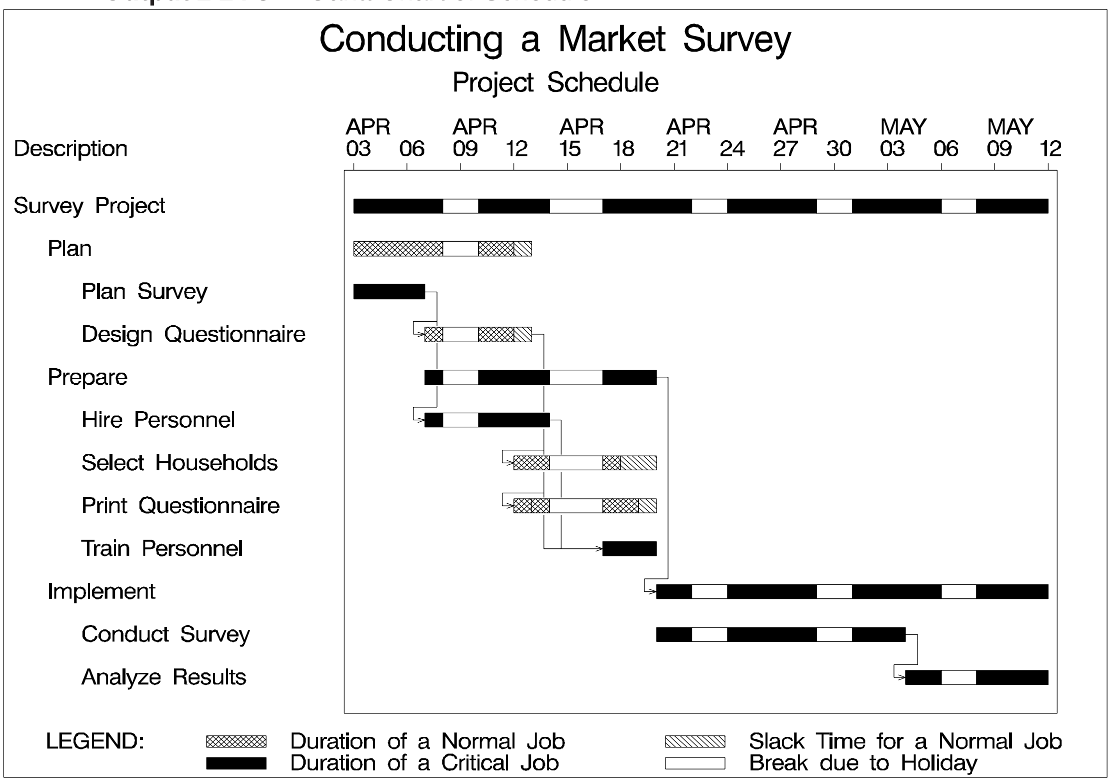
```
                         Conducting a Market Survey
                       Early and Late Start Schedule

                P  P  W      a                                         d
    p           R  R  B      c                                         u
    r           O  O  S      t                                         r
    o           J  J  _      i            s         s         s        a
    j           _  _  C      v            u         u         u        t
 O  e           D  L  O      i            c         c         c        i
 b  c           U  E  D      t            c         c         c        o
 s  t           R  V  E      y            1         2         3        n

  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

                                      E                 L
                               E      _        L        _  T  F
                               _      F        _        F  _  _   E   L
                               S      I        S        I  F  F   S   S
                               T      N        T        N  L  L   _   _
 O                             A      I        A        I  O  O   A   A
 b  i                          R      S        R        S  A  A   S   S
 s  d                          T      H        T        H  T  T   C   C

  1  Survey Project        03APR95  11MAY95  03APR95  11MAY95  0  0   0   0
  2  Plan                  03APR95  11APR95  03APR95  12APR95  1  1   1   1
  3  Plan Survey           03APR95  06APR95  03APR95  06APR95  0  0   2   2
  4  Design Questionnaire  07APR95  11APR95  10APR95  12APR95  1  0   3   3
  5  Prepare               07APR95  19APR95  07APR95  19APR95  0  0   4   4
  6  Hire Personnel        07APR95  13APR95  07APR95  13APR95  0  0   5   5
  7  Select Households     12APR95  17APR95  17APR95  19APR95  2  2   6   8
  8  Print Questionnaire   12APR95  18APR95  13APR95  19APR95  1  1   7   6
  9  Train Personnel       17APR95  19APR95  17APR95  19APR95  0  0   8   7
 10  Implement             20APR95  11MAY95  20APR95  11MAY95  0  0   9   9
 11  Conduct Survey        20APR95  03MAY95  20APR95  03MAY95  0  0  10  10
 12  Analyze Results       04MAY95  11MAY95  04MAY95  11MAY95  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, you 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 2.24.3. 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 f=swiss 'Conducting a Market Survey';
title2 c=black f=swiss h=1.5 'Project Schedule';

proc gantt graphics data=gant holidata=holidata;
   chart / holiday=(hol)
           interval=weekday
           font=swiss skip=2 height=1.2
           nojobnum
           compress noextrange
           activity=activity succ=(succ1-succ3)
           cprec=cyan cmile=magenta
           caxis=black cframe=ligr;
   id   id;
run;
```

Output 2.24.3. 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 2.24.4).

```
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 2.24.4. Survey Project Summary

```
                                 Market Survey
                                Summary Schedule

activity   PROJ_DUR  duration  E_START  E_FINISH  L_START  L_FINISH  T_FLOAT

Plan           7         6     03APR95  11APR95   03APR95  12APR95      1
Prepare        8         8     07APR95  19APR95   07APR95  19APR95      0
Implement     16        18     20APR95  11MAY95   20APR95  11MAY95      0
```

The variable WBS_CODE in the Schedule data set (see Output 2.24.2) 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 below.

```
   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 2.24.5. Subproject Schedule

```
                                 Market Survey
                              Sub-Project Schedule

   a           P P   W      a     d              E               L
   c        p  R R   B      c     u       E      _       L       _ T
   t        r  O O   S      t     r       _      F       _       F _
   i        o  J J   _      i     a       S      I       S       I F
   v        j  _ _   C      v     t       T      N       T       N L
   i        e  D L   O      i     i       A      I       A       I O
   t        c  U E   D      t     o       R      S       R       S A
   y        t  R V   E      y     n       T      H       T       H T

 Prepare  Survey  8 1 0.1   Prepare  8 07APR95 19APR95 07APR95 19APR95 0
 hire per Prepare . 2 0.1.0 hire per 5 07APR95 13APR95 07APR95 13APR95 0
 select h Prepare . 2 0.1.2 select h 3 12APR95 17APR95 17APR95 19APR95 2
 print q  Prepare . 2 0.1.3 print q  4 12APR95 18APR95 13APR95 19APR95 1
 trn per  Prepare . 2 0.1.1 trn per  3 17APR95 19APR95 17APR95 19APR95 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 15, 1995. The following program schedules the project with a *project start and finish* specification. The resulting summary schedule for the subprojects is shown in Output 2.24.6. Note that the late finish time of the project is the 12th of May because there is a weekend on the 13th and 14th of May, 1995.

```
proc cpm data=survey date='3apr95'd out=survout2
         interval=weekday holidata=holidata
         fbdate='15may95'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 2.24.6. 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	03APR95	11APR95	04APR95	13APR95	2
Prepare	8	8	07APR95	19APR95	10APR95	20APR95	1
Implement	16	18	20APR95	11MAY95	21APR95	12MAY95	1

Note that 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 2.24.7: all the subprojects have T_FLOAT = '0'.

Output 2.24.7. 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	03APR95	11APR95	03APR95	11APR95	0
Prepare	8	8	07APR95	19APR95	07APR95	19APR95	0
Implement	16	18	20APR95	11MAY95	20APR95	11MAY95	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. Note that 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

$$\text{E\_START} + \text{duration}$$

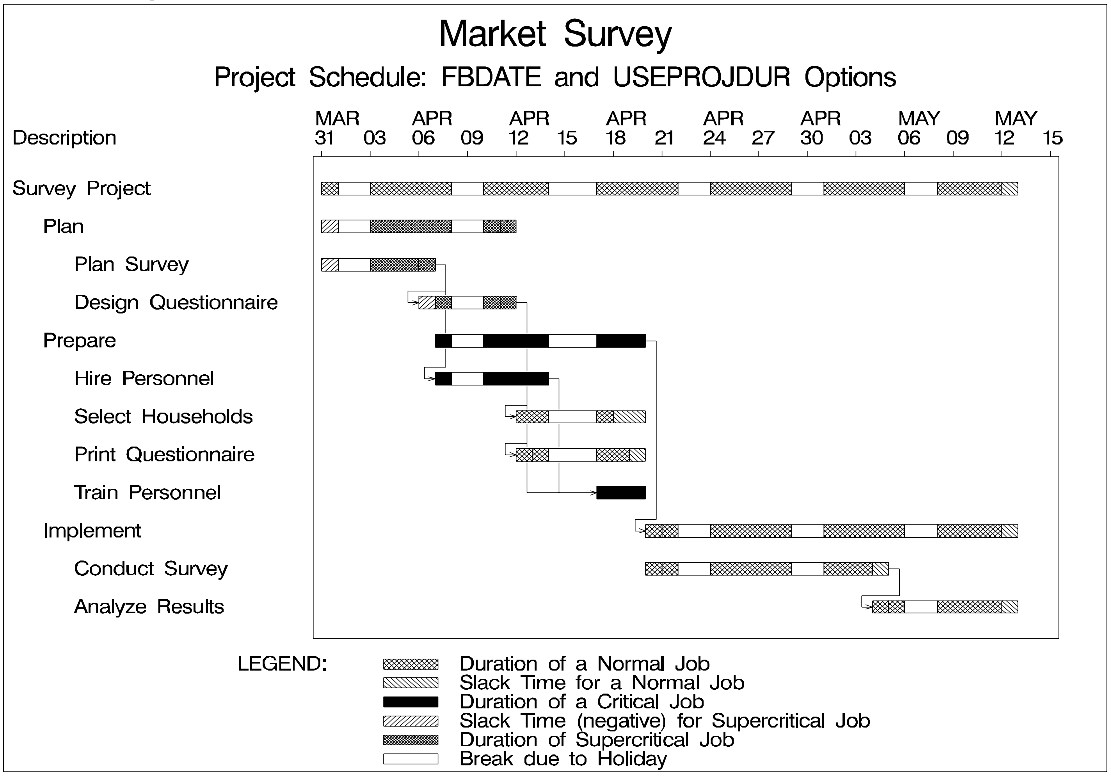
and this value is used during the backward pass.

The summary schedule resulting from the use of the USEPROJDUR option is shown in Output 2.24.8. Note the difference in the schedules in Output 2.24.7 and Output 2.24.8. In Output 2.24.7, the *computed project duration*, PROJ_DUR, is used to set an upper bound on the late finish time of each subproject, while in Output 2.24.8, 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 2.24.9.

Output 2.24.8. 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     03APR95  11APR95   31MAR95  10APR95      -1
Prepare        8         8     07APR95  19APR95   07APR95  19APR95       0
Implement     16        18     20APR95  11MAY95   21APR95  12MAY95       1
```

Output 2.24.9. 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 10, 1995, 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 2.24.10.

```
data survey2;
   format aldate date7.;
   set survey;
   if activity="Implement" then do;
      altype="fle";
      aldate='10may95'd;
      end;

   run;

proc cpm data=survey2 date='3apr95'd out=survout5
         interval=weekday holidata=holidata
         fbdate='15jun95'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 2.24.10. USEPROJDUR option and Alignment Date

```
                              Market Survey
          Summary Schedule: USEPROJDUR option and Alignment date

activity   PROJ_DUR  duration  E_START  E_FINISH  L_START  L_FINISH  T_FLOAT

Plan           7         6     03APR95  11APR95   31MAR95  10APR95      -1
Prepare        8         8     07APR95  19APR95   06APR95  18APR95      -1
Implement     16        18     20APR95  11MAY95   19APR95  10MAY95      -1
```

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 2.28. Statements and Options Specified in Examples 2.1 – 2.12

Statement	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
ACTIVITY	X					X	X	X	X	X	X	X
ALIGNDATE												X
ALIGNTYPE												X
CALID										X		
DURATION	X	X	X	X	X	X	X	X	X	X	X	X
HEADNODE		X	X	X	X							
HOLIDAY								X	X	X		
ID	X	X	X	X	X							
SUCCESSOR	X					X	X	X	X	X	X	X
TAILNODE		X	X	X	X							
Option	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
ALAGCAL=											X	
CALENDAR=									X	X	X	
COLLAPSE											X	
DATA=	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
DAYLENGTH=							X		X	X		
DAYSTART=							X					
FBDATE=			X									
HOLIDATA=								X	X	X		

Table 2.28. (continued)

Statement	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
HOLIDUR=								X		X		
HOLIFIN=								X	X	X		
INTERVAL=			X	X	X	X	X	X	X		X	X
LAG=											X	
OUT=		X		X	X	X		X	X	X	X	
WORKDAY=									X	X		
XFERVARS												X

Table 2.29. Statements and Options Specified in Examples 2.13 – 2.22

Statement	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
ACTIVITY	X						X	X		X	X	X
ACTUAL	X											
ALIGNDATE												X
ALIGNTYPE												X
BASELINE	X					X						
DURATION	X	X	X	X	X	X	X	X	X	X	X	X
HEADNODE		X	X	X	X	X			X			
HOLIDAY	X	X	X	X	X	X		X				X
ID		X	X	X	X	X			X		X	X
PROJECT												X
RESOURCE		X	X	X	X	X	X	X		X	X	
SUCCESSOR	X						X	X		X	X	X
TAILNODE		X	X	X	X	X			X			
Option	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
A_FINISH=	X											
A_START=	X											
ACTDELAY=						X						
ADDCAL											X	
ADDWBS												X
AUTOUPDT	X											
AVPROFILE					X	X	X	X				
CALENDAR=											X	
COLLAPSE							X	X				
COMPARE=	X					X						
CUMUSAGE				X	X	X						
DATA=	X	X	X	X	X	X	X	X	X	X	X	X
DATE=	X	X	X	X	X	X	X	X	X		X	X
DELAY=				X	X			X				
DELAYANALYSIS			X	X	X	X		X				
FBDATE=												X
F_FLOAT							X					

Table 2.29. (continued)

Statement	13	14	15	16	17	18	19	20	21	22	23	24
HOLIDATA=	X	X	X	X	X	X	X	X				X
HOLIFIN=	X											
INFEASDIAGNOSTIC					X	X						
INTERVAL=		X	X	X	X	X	X	X			X	X
MAXDATE=		X										
MINSEGMTDUR=							X					
NOAUTOUPDT	X											
OBSTYPE=			X	X	X	X	X	X			X	
OUT=	X		X	X	X	X	X	X	X	X	X	X
ORDERALL												X
PCTCOMP=	X											
PERIOD=			X	X	X	X	X	X		X	X	
RCPROFILE					X	X	X	X				
REMDUR=	X											
RESID=								X				
RESOURCEIN=			X	X	X	X	X	X		X	X	
RESOURCEOUT=		X	X	X	X	X	X	X			X	
RESOURCESCHED=											X	
ROUTNOBREAK				X								
RSCHEDID=											X	
SCHEDRULE=			X									
SET=	X					X						
SEPCRIT												X
SHOWFLOAT	X											
STOPDATE=										X		
T_FLOAT						X	X					
TIMENOW=	X											
USEPROJDUR												X
WORK=											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 and Sons, Inc.

Horowitz, E. and Sahni, S. (1976), *Fundamentals of Data Structures*, Potomac, MD: Computer Science Press, Inc.

Kulkarni, R. (1991), "Scheduling with the CPM Procedure," *Proceedings of the Sixteenth Annual SAS Users Group International Conference*.

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*, Vol. 7, No. 5, 646-669.

Minieka, E. (1978), *Optimization Algorithms for Networks and Graphs*, New York: Marcel Dekker, Inc.

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

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*, Vol 11, No. 5, 839-860.

Wiest, J.D. (1967), “A Heuristic Model for Scheduling Large Projects with Limited Resources,” *Management Science*, 13, 359-377.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/OR User's Guide: Project Management, Version 8,* Cary, NC: SAS Institute Inc., 1999. 806 pp.

SAS/OR User's Guide: Project Management, Version 8

ISBN 1–58025–492–6

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999