SAS/OR® 12.1 User's Guide
Bill of Material Processing

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2012. *SAS/OR® 12.1 User's Guide: Bill of Material Processing*. Cary, NC: SAS Institute Inc.

SAS/OR® 12.1 User's Guide: Bill of Material Processing

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

Electronic book 1, August 2012

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

Contents

Acknowledgments

Credits

Documentation

Writing	Tien-yi D. Shaw, Michelle Opp
Editing	Anne Baxter, Virginia Clark, Ed Huddleston
Documentation Support	Tim Arnold, Melanie Gratton, Michelle Opp, Daniel Underwood
Technical Review	Marc-david Cohen, Phil Gibbs, Tao Huang, Edward P. Hughes, Charles B. Kelly, Michelle Opp, Bengt Pederson

Software

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

In the following list, the name of the developer currently supporting the component is listed first. Other developers previously worked on the procedure.

PROC BOM	Tien-yi D. Shaw
BOM Macros	Tien-yi D. Shaw
BOM Web Examples	Yi Liao, Tien-yi D. Shaw

Support Groups

Software Testing	Bengt Pederson, Tao Huang
Technical Support	Tonya Chapman

Acknowledgments

Many people have been instrumental in the development of SAS/OR Bill of Material software. We would like to acknowledge Dave Jennings of Lockheed Martin, who has been especially helpful.

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

Chapter 1
What's New in SAS/OR 12.1

Contents

Overview

SAS/OR 12.1 delivers a broad range of new capabilities and enhanced features, encompassing optimization, constraint programming, and discrete-event simulation. SAS/OR 12.1 enhancements significantly improve performance and expand your tool set for building, analyzing, and solving operations research models.

In previous years, SAS/OR® software was updated only with new releases of Base SAS® software, but this is no longer the case. This means that SAS/OR software can be released to customers when enhancements are ready, and the goal is to update SAS/OR every 12 to 18 months. To mark this newfound independence, the release numbering scheme for SAS/OR is changing with this release. This new numbering scheme will be maintained when new versions of Base SAS and SAS/OR ship at the same time. For example, when Base SAS 9.4 is released, SAS/OR 13.1 will be released.

Highlights of Enhancements in SAS/OR 12.1

Highlights of the SAS/OR enhancements include the following:

- multithreading is used to improve performance in these three areas:

  - PROC OPTMODEL problem generation
  - multistart for nonlinear optimization
  - option tuning for mixed integer linear optimization
- concurrent solve capability (experimental) for linear programming (LP) and nonlinear programming (NLP)
- improvements to all simplex LP algorithms and mixed integer linear programming (MILP) solver
- new decomposition (DECOMP) algorithm for LP and MILP
- new option for controlling MILP cutting plane strategy
- new conflict search capability for MILP solver
- option tuning for PROC OPTMILP
- new procedure, PROC OPTNET, for network optimization and analysis
- new SUBMIT block for invoking SAS code within PROC OPTMODEL
- SAS Simulation Studio improvements:
  - one-click connection of remote blocks in large models
  - autoscrolling for navigating large models
  - new search capability for block types and label content
  - alternative Experiment window configuration for large experiments
  - selective animation capability
  - new submodel component (experimental)

The CLP Procedure

In SAS/OR 12.1, the CLP procedure adds two classes of constraints that expand its capabilities and can accelerate its solution process. The LEXICO statement imposes a lexicographic ordering between pairs of variable lists. Lexicographic order is essentially analogous to alphabetical order but expands the concept to include numeric values. One vector (list) of values is lexicographically less than another if the corresponding elements are equal up to a certain point and immediately after that point the next element of the first vector is numerically less than the second. Lexicographic ordering can be useful in eliminating certain types of symmetry that can arise among solutions to constraint satisfaction problems (CSPs). Imposing a lexicographic ordering eliminates many of the mutually symmetric solutions, reducing the number of permissible solutions to the problem and in turn shortening the solution process.

Another constraint class that is added to PROC CLP for SAS/OR 12.1 is the bin-packing constraint, imposed via the PACK statement. A bin-packing constraint directs that a specified number of items must be placed into a specified number of bins, subject to the capacities (expressed in numbers of items) of the bins. The PACK statement provides a compact way to express such constraints, which can often be useful components of larger CSPs or optimization problems.

The DTREE, GANTT, and NETDRAW Procedures

In SAS/OR 12.1 the DTREE, GANTT, and NETDRAW procedures each add procedure-specific graph styles that control fonts, line colors, bar and node fill colors, and background images.

Supporting Technologies for Optimization

The underlying improvements in optimization in SAS/OR 12.1 are chiefly related to multithreading, which denotes the use of multiple computational cores to enable computations to be executed in parallel rather than serially. Multithreading can provide dramatic performance improvements for optimization because these underlying computations are performed many times in the course of an optimization process.

The underlying linear algebra operations for the linear, quadratic, and nonlinear interior point optimization algorithms are now multithreaded. The LP, QP, and NLP solvers can be used by PROC OPTMODEL, PROC OPTLP, and PROC OPTQP in SAS/OR. For nonlinear optimization with PROC OPTMODEL, the evaluation of nonlinear functions is multithreaded for improved performance.

Finally, the process of creating an optimization model from PROC OPTMODEL statements has been multithreaded. PROC OPTMODEL contains powerful declarative and programming statements and is adept at enabling data-driven definition of optimization models, with the result that a rather small section of PROC OPTMODEL code can create a very large optimization model when it is executed. Multithreading can dramatically shorten the time that is needed to create an optimization model.

In SAS/OR 12.1 you can use the NTHREADS= option in the PERFORMANCE statement in PROC OPTMODEL and other SAS/OR optimization procedures to specify the number of cores to be used. Otherwise, SAS detects the number of cores available and uses them.

PROC OPTMODEL: Nonlinear Optimization

The nonlinear optimization solver that PROC OPTMODEL uses builds on the introduction of multithreading for its two most significant improvements in SAS/OR 12.1. First, in addition to the nonlinear solver options ALGORITHM=ACTIVESET and ALGORITHM=INTERIORPOINT, SAS/OR 12.1 introduces the ALGORITHM=CONCURRENT option (experimental), with which you can invoke both the active set and interior point algorithms for the specified problem, running in parallel on separate threads. The solution process terminates when either of the algorithms terminates. For repeated solves of a number of similarly structured problems or simply for problems for which the best algorithm isn't readily apparent, ALGORITHM=CONCURRENT should prove useful and illuminating.

Second, multithreading is central to the nonlinear optimization solver's enhanced multistart capability, which now takes advantage of multiple threads to execute optimizations from multiple starting points in parallel. The multistart capability is essential for problems that feature nonconvex nonlinear functions in either or both of the objective and the constraints because such problems might have multiple locally optimal points. Starting optimization from several different starting points helps to overcome this difficulty, and multithreading this process helps to ensure that the overall optimization process runs as fast as possible.

Linear Optimization with PROC OPTMODEL and PROC OPTLP

Extensive improvements to the primal and dual simplex linear optimization algorithms produce better performance and better integration with the crossover algorithm, which converts solutions that are found by the interior point algorithm into more usable basic optimal solutions. The crossover algorithm itself has undergone extensive enhancements that improve its speed and stability.

Paralleling developments in nonlinear optimization, SAS/OR 12.1 linear optimization introduces a concurrent algorithm, invoked with the ALGORITHM=CONCURRENT option, in the SOLVE WITH LP statement for PROC OPTMODEL or in the PROC OPTLP statement. The concurrent LP algorithm runs a selection of linear optimization algorithms in parallel on different threads, with settings to suit the problem at hand. The optimization process terminates when the first algorithm identifies an optimal solution. As with nonlinear optimization, the concurrent LP algorithm has the potential to produce significant reductions in the time needed to solve challenging problems and to provide insights that are useful when you solve a large number of similarly structured problems.

Mixed Integer Linear Optimization with PROC OPTMODEL and PROC OPTMILP

Mixed integer linear optimization in SAS/OR 12.1 builds on and extends the advances in linear optimization. Overall, solver speed has increased by over 50% (on a library of test problems) compared to SAS/OR 9.3. The branch-and-bound algorithm has approximately doubled its ability to evaluate and solve component linear optimization problems (which are referred to as nodes in the branch-and-bound tree). These improvements have significantly reduced solution time for difficult problems.

The Decomposition Algorithm

The most fundamental change to both linear and mixed integer linear optimization in SAS/OR 12.1 is the addition of the decomposition (DECOMP) algorithm, which is invoked with a specialized set of options in the SOLVE WITH LP and SOLVE WITH MILP statements for PROC OPTMODEL or in the DECOMP statement for PROC OPTLP and PROC OPTMILP. For many linear and mixed integer linear optimization problems, most of the constraints apply only to a small set of decision variables. Typically there are many such sets of constraints, complemented by a small set of linking constraints that apply to all or most of the decision variables. Optimization problems with these characteristics are said to have a "block-angular" structure, because it is easy to arrange the rows of the constraint matrix so that the nonzero values, which correspond to the local sets of constraints, appear as blocks along the main diagonal.

The DECOMP algorithm exploits this structure, decomposing the overall optimization problem into a set of component problems that can be solved in parallel on separate computational threads. The algorithm repeatedly solves these component problems and then cycles back to the overall problem to update key information that is used the next time the component problems are solved. This process repeats until it produces a solution to the complete problem, with the linking constraints present. The combination of parallelized solving of the component problems and the iterative coordination with the solution of the overall

problem can greatly reduce solution time for problems that were formerly regarded as too time-consuming to solve practically.

To use the DECOMP algorithm, you must either manually or automatically identify the blocks of the constraint matrix that correspond to component problems. The METHOD= option controls the means by which blocks are identified. METHOD=USER enables you to specify the blocks yourself, using the .block suffix to declare blocks. This is by far the most common method of defining blocks. If your problem has a significant or dominant network structure, you can use METHOD=NETWORK to identify the blocks in the problem automatically. Finally, if no linking constraints are present in your problem, then METHOD=AUTO identifies the blocks automatically.

The DECOMP algorithm uses a number of detailed options that specify how the solution processes for the component problems and the overall problem are configured and how they coordinate with each other. You can also specify the number of computational threads to make available for processing component problems and the level of detail in the information to appear in the SAS log. Options specific to the linear and mixed integer linear solvers that are used by the DECOMP algorithm are largely identical to those for the respective solvers.

Setting the Cutting Plane Strategy

Cutting planes are a major component of the mixed integer linear optimization solver, accelerating its progress by removing fractional (not integer feasible) solutions. SAS/OR 12.1 adds the CUTSTRATEGY= option in the PROC OPTMILP statement and in the SOLVE WITH MILP statement for PROC OPTMODEL, enabling you to determine the aggressiveness of your overall cutting plane strategy. This option complements the individual cut class controls (CUTCLQUE=, CUTGOMORY=, CUTMIR=, and so on), with which you can enable or disable certain cut types, and the ALLCUTS= option, which enables or disables all cutting planes. In contrast, the CUTSTRATEGY= option controls cuts at a higher level, creating a profile for cutting plane use. As the cut strategy becomes more aggressive, more effort is directed toward creating cutting planes and more cutting planes are applied. The available values of the CUTSTRATEGY= option are AUTOMATIC, BASIC, MODERATE, and AGGRESSIVE; the default is AUTOMATIC. The precise cutting plane strategy that corresponds to each of these settings can vary from problem to problem, because the strategy is also tuned to suit the problem at hand.

Conflict Search

Another means of accelerating the solution process for mixed integer linear optimization takes information from infeasible linear optimization problems that are encountered during an initial exploratory phase of the branch-and-bound process. This information is analyzed and ultimately is used to help the branch-and-bound process avoid combinations of decision variable values that are known to lead to infeasibility. This approach, known as conflict analysis or conflict search, influences presolve operations on branch-and-bound nodes, cutting planes, computation of decision variable bounds, and branching. Although the approach is complex, its application in SAS/OR 12.1 is straightforward. The CONFLICTSEARCH= option in the PROC OPTMILP statement or the SOLVE WITH MILP statement in PROC OPTMODEL enables you to specify the level of conflict search to be performed. The available values for the CONFLICTSEARCH=

option are NONE, AUTOMATIC, MODERATE, and AGGRESSIVE. A more aggressive search strategy explores more branch-and-bound nodes initially before the branch-and-bound algorithm is restarted with information from infeasible nodes included. The default value is AUTOMATIC, which enables the solver to choose the search strategy.

PROC OPTMILP: Option Tuning

The final SAS/OR 12.1 improvement to the mixed integer linear optimization solver is option tuning, which helps you determine the best option settings for PROC OPTMILP. There are many options and settings available, including controls on the presolve process, branching, heuristics, and cutting planes. The TUNER statement enables you to investigate the effects of the many possible combinations of option settings on solver performance and determine which should perform best. The PROBLEMS= option enables you to submit several problems for tuning at once. The OPTIONMODE= option specifies the options to be tuned. OPTIONMODE=USER indicates that you will supply a set of options and initial values via the OPTIONVALUES= data set, OPTIONMODE=AUTO (the default) tunes a small set of predetermined options, and OPTIONMODE=FULL tunes a much more extensive option set.

Option tuning starts by using an initial set of option values to solve the problem. The problem is solved repeatedly with different option values, with a local search algorithm to guide the choices. When the tuning process terminates, the best option values are output to a data set specified by the SUMMARY= option. You can control the amount of time used by this process by specifying the MAXTIME= option. You can multithread this process by using the NTHREADS= option in the PERFORMANCE statement for PROC OPTMILP, permitting analyses of various settings to occur simultaneously.

PROC OPTMODEL: The SUBMIT Block

In SAS/OR 12.1, PROC OPTMODEL adds the ability to execute other SAS code nested inside PROC OPTMODEL syntax. This code is executed immediately after the preceding PROC OPTMODEL syntax and before the syntax that follows. Thus you can use the SUBMIT block to, for example, invoke other SAS procedures to perform analyses, to display results, or for other purposes, as an integral part of the process of creating and solving an optimization model with PROC OPTMODEL. This addition makes it even easier to integrate the operation of PROC OPTMODEL with other SAS capabilities.

To create a SUBMIT block, use a SUBMIT statement (which must appear on a line by itself) followed by the SAS code to be executed, and terminate the SUBMIT block with an ENDSUBMIT statement (which also must appear on a line by itself). The SUBMIT statement enables you to pass PROC OPTMODEL parameters, constants, and evaluated expressions to the SAS code as macro variables.

Network Optimization with PROC OPTNET

PROC OPTNET, new in SAS/OR 12.1, provides several algorithms for investigating the characteristics of networks and solving network-oriented optimization problems. A network, sometimes referred to as a graph,

consists of a set of nodes that are connected by a set of arcs, edges, or links. There are many applications of network structures in real-world problems, including supply chain analysis, communications, transportation, and utilities problems. PROC OPTNET addresses the following classes of network problems:

- biconnected components
- maximal cliques
- connected components
- cycle detection
- weighted matching
- minimum-cost network flow
- minimum cut
- minimum spanning tree
- shortest path
- transitive closure
- traveling salesman

PROC OPTNET syntax provides a dedicated statement for each problem class in the preceding list.

The formats of PROC OPTNET input data sets are designed to fit network-structured data, easing the process of specifying network-oriented problems. The underlying algorithms are highly efficient and can successfully address problems of varying levels of detail and scale. PROC OPTNET is a logical destination for users who are migrating from some of the legacy optimization procedures in SAS/OR. Former users of PROC NETFLOW can turn to PROC OPTNET to solve shortest-path and minimum-cost network flow problems, and former users of PROC ASSIGN can instead use the LINEAR_ASSIGNMENT statement in PROC OPTNET to solve assignment problems.

SAS Simulation Studio 12.1

SAS Simulation Studio 12.1, a component of SAS/OR 12.1 for Windows environments, adds several features that improve your ability to build, explore, and work with large, complex discrete-event simulation models. Large models present a number of challenges to a graphical user interface such as that of SAS Simulation Studio. Connection of model components, navigation within a model, identification of objects or areas of interest, and management of different levels of modeling are all tasks that can become more difficult as the model size grows significantly beyond what can be displayed on one screen. An indirect effect of model growth is an increased number of factors and responses that are needed to parameterize and investigate the performance of the system being modeled.

Improvements in SAS Simulation Studio 12.1 address each of these issues. In SAS Simulation Studio, you connect blocks by dragging the cursor to create links between output and input ports on regular blocks and Connector blocks. SAS Simulation Studio 12.1 automatically scrolls the display of the Model window as

you drag the link that is being created from its origin to its destination, thus enabling you to create a link between two blocks that are located far apart (additionally you can connect any two blocks by clicking on the OutEntity port of the first block and then clicking on the InEntity port of the second block). Automatic scrolling also enables you to navigate a large model more easily. To move to a new area in the Model window, you can simply hold down the left mouse button and drag the visible region of the model to the desired area. This works for simple navigation and for moving a block to a new, remote location in the model.

SAS Simulation Studio 12.1 also enables you to search among the blocks in a model and identify the blocks that have a specified type, a certain character string in their label, or both. From the listing of identified blocks, you can open the Properties dialog box for each identified block and edit its settings. Thus, if you can identify a set of blocks that need similar updates, then you can make these updates without manually searching through the model for qualifying blocks and editing them individually. For very large models, this capability not only makes the update process easier but also makes it more thorough because you can identify qualifying blocks centrally.

When you design experiments for large simulation models, you often need a large number of factors to parameterize the model and a large number of responses to track system performance in sufficient detail. This was a challenge prior to SAS Simulation Studio 12.1 because the Experiment window displayed factors and responses in the header row of a table, with design points and their replications' results displayed in the rows below. A very large number of factors and responses did not fit on one screen in this display scheme, and you had to scroll across the Experiment window to view all of them.

SAS Simulation Studio 12.1 provides you with two alternative configurations for the Experiment window. The Design Matrix tab presents the tabular layout described earlier. The Design Point tab presents each design point in its own display. Factors and responses (summarized over replications) are displayed in separate tables, each with the factor or response names appearing in one column and the respective values in a second column. This layout enables a large number of factors and responses to be displayed. Response values for each replication of the design point can be displayed in a separate window.

SAS Simulation Studio 12.1 enhances its multilevel model management features by introducing the submodel component (experimental). Like the compound block, the submodel encapsulates a group of SAS Simulation Studio blocks and their connections, but the submodel outpaces the compound block in some important ways. The submodel, when expanded, opens in its own window. This means a submodel in its collapsed form can be placed close to other blocks in the Model window without requiring space for its expanded form (as is needed for compound blocks). The most important property of the submodel is its ability to be copied and instantiated in several locations simultaneously, whether in the same model, in different models in the same project, or in different projects. Each such instance is a direct reference to the original submodel, not a disconnected copy. Thus you can edit the submodel by editing any of its instances; changes that are made to any instance are propagated to all current and future instances of the submodel. This feature enables you to maintain consistency across your models and projects.

Finally, SAS Simulation Studio 12.1 introduces powerful new animation controls that should prove highly useful in debugging simulation models. In the past, animation could be switched on or off and its speed controlled, but these choices were made for the entire model. If you needed to animate a particular segment of the model, perhaps during a specific time span for the simulation clock, you had to focus your attention on that area and pay special attention when the time period of interest arrived. In SAS Simulation Studio 12.1 you can select both the area of the model to animate (by selecting a block or a compound block) and the time period over which animation should occur (by specifying the start and end times for animation). You can also control simulation speed for each such selection. Multiple selections are supported so that you can choose to animate several areas of the model, each during its defined time period and at its chosen speed.

Chapter 2
Using This Book

Contents

Purpose

SAS/OR User's Guide: Bill of Material Processing provides a complete reference for the bill of material processing (BOM) tools in SAS/OR software. This book serves as the primary documentation for the BOM procedure and the bill of material postprocessing macros. Chapter 3 describes the BOM procedure, and Chapter 4 describes the bill of material postprocessing macros.

"Using This Book" describes the organization of this book and the conventions used in the text and example code. To gain full benefit from using this book, you should familiarize yourself with the information presented in this section and refer to it when needed. The section "Additional Documentation for SAS/OR Software" on page 12 refers to other documents that contain related information.

Organization

Chapter 3 describes the BOM procedure. Chapter 4 and Appendix A describe the bill of material postprocessing macros and a BOM Web example, respectively. Each chapter description is self-contained; you need to be familiar with only the basic features of the SAS System and SAS terminology to use most of them. The following list summarizes the types of information provided for the BOM procedure:

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

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

Syntax constitutes the major reference section for the syntax of the procedure. First, the statement syntax is summarized. Next, a functional summary table lists all the statements and options in the procedure, classified by function. In addition, the online version includes a Dictionary of Options, which provides an alphabetical list of all options in the BOM procedure. The PROC BOM statement is then described, followed by a description of the STRUCTURE statement.

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

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

References lists references that are relevant to the chapter.

Typographical Conventions

The printed version of *SAS/OR User's Guide: Bill of Material Processing* uses various type styles, as explained by the following list:

roman	is the standard type style used for most text.

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

Conventions for Examples

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

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

In addition, the graphics output shown in Chapter 3 is produced with the following options:

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

The remaining graphics options used in this book depend on the example that creates the output. The general options for the graphics output of Example 3.1 are as follows:

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

In addition, the following PATTERN statement is used to create the color output for this example:

```
pattern1 v=s c=blue;
```

The general options for the graphics output of Example 3.2 are as follows:

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

In addition, the following PATTERN statement is used to create the color output for this example:

```
pattern1 v=e c=blue;
```

Accessing the SAS/OR Sample Library

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

Online Documentation

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

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

Additional Documentation for SAS/OR Software

In addition to *SAS/OR User's Guide: Bill of Material Processing*, you might find the following documents helpful when using SAS/OR software:

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

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

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

SAS/OR User's Guide: Mathematical Programming Examples

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

SAS/OR User's Guide: Mathematical Programming Legacy Procedures

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

SAS/OR User's Guide: Network Optimization Algorithms

provides documentation for a set of algorithms that can be used to investigate the characteristics of networks and to solve network-oriented optimization problems. This book also documents PROC OPTNET, which invokes these algorithms and provides network-structured formats for input and output data.

SAS/OR User's Guide: Project Management

provides documentation for the project management procedures in SAS/OR software. This book serves as the primary documentation for the CPM, DTREE, GANTT, NETDRAW, and PM procedures, the earned value management macros, the Microsoft Project conversion macros, and the PROJMAN application.

SAS/OR Software: Project Management Examples, Version 6

contains a series of examples that illustrate how to use SAS/OR software to manage projects. Each chapter contains a complete project management scenario and describes how to use PROC GANTT, PROC CPM, and PROC NETDRAW, in addition to other reporting and graphing procedures in the SAS System, to perform the necessary project management tasks.

SAS Simulation Studio: User's Guide

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

Chapter 3
The BOM Procedure

Contents

Overview: BOM Procedure

The BOM procedure performs bill of material processing. It reads all product structure records from a product structure data file and all part "master" records from a part master file, and composes the combined

information into indented bill of material. In addition, PROC BOM can also output a summarized parts list, which lists all items and their total quantities required (needed to be made or ordered) in order to fill a given production plan. A *production plan* is an agreed-upon plan that comes from the aggregate planning function; specifically, it is the overall level of manufacturing output planned to be produced for each product family in each planning period (Cox and Blackstone 1998). Production plan information can be included in the part master file.

According to Cox and Blackstone (1998), a *product structure record* defines the relationship of one component to its immediate parent. It also contains fields for quantity required, scrap factor, lead-time offset, engineering effectivity, and so on. A *part master record* typically contains identifying and descriptive data such as part number and part description, and control values (for instance, lead time, lot size, cost, etc.). It may contain "dynamic" data such as inventory status (for instance, quantities on hand) and production plan data (for instance, requirements, due dates, etc.). Part master records are linked by product structure records or bill of material records, thus defining the bill of material.

Also based on Cox and Blackstone (1998), a *bill of material* (BOM) for an item is a list of all items, ingredients, or materials needed to make one production run of the given item. The bill of material may also be called the *formula*, *recipe*, or *ingredients list* in certain process industries. The way in which the bill of material data are organized and presented is called the *structure* of the bill of material or the structure of the product.

A variety of display formats is available for bill of material. The simplest format is the *single-level bill of material*. It consists of a list of all components that are directly used in a parent item. It shows only the relationships one level down. On the other hand, a *multilevel bill of material* provides a display of all components that are directly or indirectly used in a parent item. When an item is a subcomponent, blend, intermediate, etc., then all of its components are also exhibited, down to purchased parts and raw materials.

An *indented bill of material* is one form of a multilevel BOM. It exhibits the highest level parent item as level 0, and all components going into this parent item are indented as level 1. All subsequent components are indented one level below their parent items; that is, the level number is increased by 1. If an item is used in more than one parent within a given product structure, it appears more than once, under every subassembly in which it is used. You can view an indented bill of material as a *family tree* with multiple levels. The *summarized bill of material* is another form of a multilevel bill of material. It lists all the items and their quantities that are used in a given product structure. Unlike the indented bill of material, it does not list the level numbers of items and does not illustrate the parent-component relationships. Moreover, the summarized bill of material lists each item only once for the total quantity required. Refer to Cox and Blackstone (1998) for further details.

The input data sets used by the procedure are the Product Structure data set and the Part Master data set. The Product Structure data set contains all product structure records, and the Part Master data set contains all part master records for a product, product line, plant, or company. You can combine the product structure data and the part master data into one input data set and assign it as the Product Structure data set; the BOM procedure assumes that all the part master information is also available in this combined data set.

The Indented BOM output data set produced by PROC BOM contains all indented bill of material for the product, product line, plant, or company. This data set is organized in such a manner that it can be easily retrieved and manipulated to generate reports, and can also be used by other SAS/OR procedures to perform additional analysis. See Example 3.1 and Example 3.2 as examples of using the NETDRAW procedure to draw a tree diagram for an indented bill of material. Chapter 4, "Bill of Material Postprocessing Macros," describes a collection of SAS macros that use the Indented BOM data set as input data to create reports or perform specialized transactions for maintaining the bill of material.

The BOM procedure can optionally produce a Summarized Parts output data set. This output data set lists all items and their quantities required (needed to be made or ordered) to fill the production plan specified in the part master file. This list is useful in gross requirements planning and other applications.

Getting Started: BOM Procedure

The ABC Lamp Company product line example from Fogarty, Blackstone, and Hoffmann (1991) illustrates the basic features of PROC BOM. The Part Master data set, PMaster0, displayed in Figure 3.1, contains the part master records for all items in the company. Each master record contains the part number (denoted by the Part variable) and the part description (denoted by the Desc variable) for an item. It also contains data on unit of measure (denoted by the Unit variable) for the item. The Product Structure data set, ParComp0, displayed in Figure 3.2, contains all product structure records in the company. Each product structure defines a parent-component relationship. The Parent variable contains the part number for the parent item and the Component variable contains the part number for the component. The QtyPer variable contains the quantity per assembly for the parent-component relationship.

The values for the Parent and Component variables are linked by the value of the Part variable in the Part Master data set to define the parent-component relationship. For example, in the 11th observation of the data set ParComp0 (as displayed in Figure 3.2), the value '1100' for the Parent variable is linked to the first observation of the data set PMaster0 (as displayed in Figure 3.1), which provides the part description and unit of measure for the parent item. Similarly, the value of '2100' for the Component variable is linked to the eighth observation of the part master file providing the master data for the component. Therefore, the parent-component relationship that is represented in the 11th observation of the product structure data set, ParComp0, should be interpreted as: each finished shaft uses 26 inches of 3/8 steel tubing.

Figure 3.1 Part Master File (PMaster0)

ABC Lamp Company

Part Master Data Set

Obs	Part	Desc	Unit
1	1100	Finished shaft	Each
2	1200	6-Diameter steel plate	Each
3	1300	Hub	Each
4	1400	1/4-20 Screw	Each
5	1500	Steel holder	Each
6	1600	One-way socket	Each
7	1700	Wiring assembly	Each
8	2100	3/8 Steel tubing	Inches
9	2200	16-Gauge lamp cord	Feet
10	2300	Standard plug terminal	Each
11	A100	Socket assembly	Each
12	B100	Base assembly	Each
13	LA01	Lamp LA	Each
14	S100	Black shade	Each

Figure 3.2 Product Structure File (ParComp0)

ABC Lamp Company

Product Structure Data Set

Obs	Parent	Component	Qty Per
1	LA01	B100	1
2	LA01	S100	1
3	LA01	A100	1
4	B100	1100	1
5	B100	1200	1
6	B100	1300	1
7	B100	1400	4
8	A100	1500	1
9	A100	1600	1
10	A100	1700	1
11	1100	2100	26
12	1500	1400	2
13	1700	2200	12
14	1700	2300	1

The following code invokes PROC BOM to produce the indented bill of material for the product '`LA01`' and the summarized parts list for the production plan in which 1 unit of '`LA01`' is planned in the current planning period (period 1).

```
  /* Create the indented BOM and summarized parts list */
proc bom data=ParComp0 pmdata=PMaster0
         out=IndBOM0 summaryout=SumBOM0;
   structure / part=Part
               parent=Parent
               component=Component
               quantity=QtyPer
               id=(Desc Unit);
run;
```

The Indented BOM data set, IndBOM0, is displayed in Figure 3.3. This data set contains the indented bill of material for the product '`LA01`'. Each record or observation in this data set contains product structure data: the part number for the parent item is contained in the _Parent_ variable, the part number for the component is contained in the _Part_ variable, and the quantity per assembly is contained in the QtyPer variable. It also contains the part master data (Desc and Unit variables) from the Part Master data set for the component identified by the _Part_ variable. If a component is used in more than one parent item, it appears in multiple records. For example, the item `1/4-20 Screw` (part number '`1400`') is used in both `Base assembly` (part number '`B100`') and `Steel holder` (part number '`1500`'); this item occurs in records identified by the values 6 and 10 for the variable Part_ID.

Figure 3.3 Indented Bill of Material (IndBOM0)

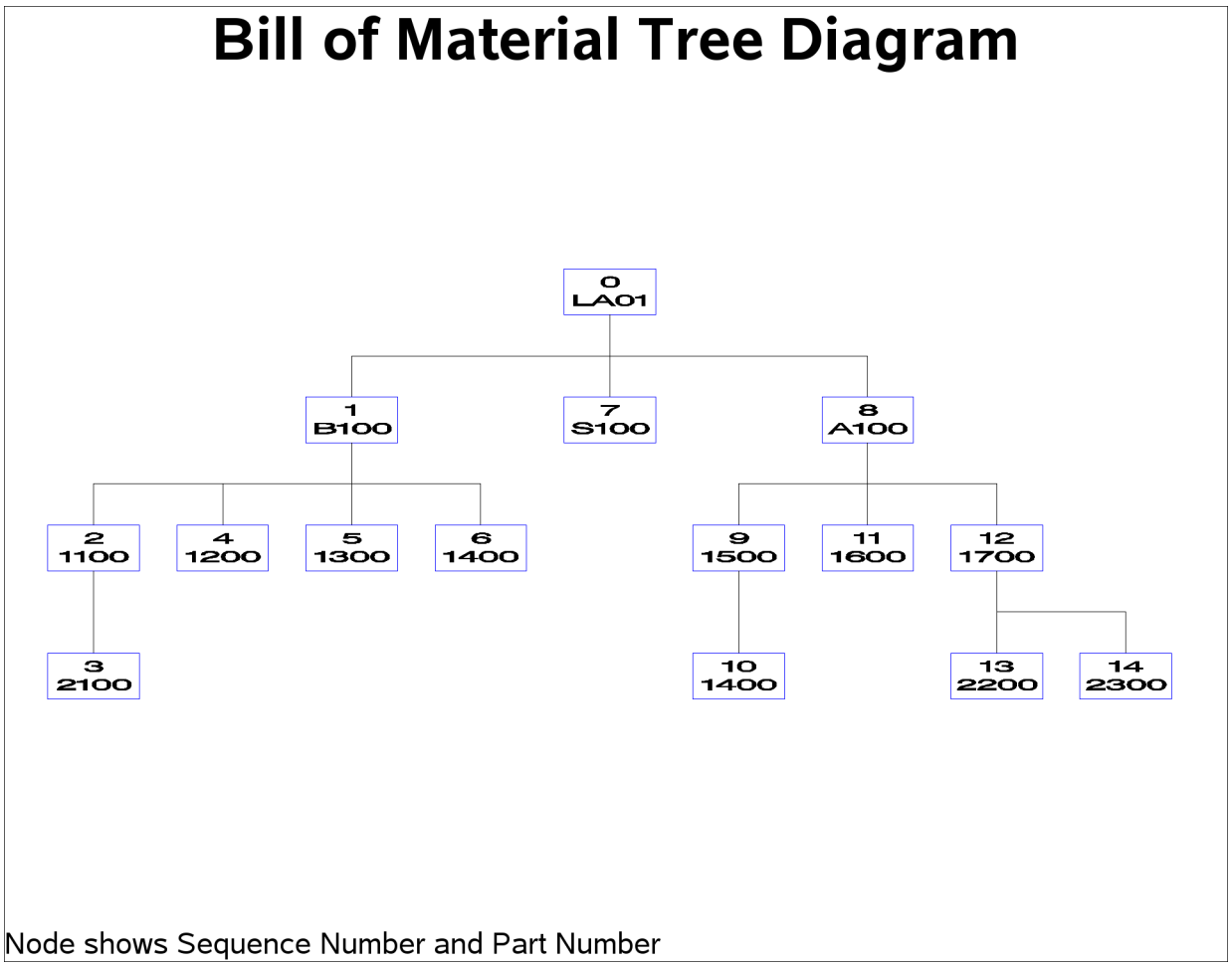
ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Desc	Qty_Per	Qty_Prod	Unit	Parent_ID	Part_ID	_Prod_
0		LA01	Lamp LA	.	1	Each	.	0	LA01
1	LA01	B100	Base assembly	1	1	Each	0	1	LA01
2	B100	1100	Finished shaft	1	1	Each	1	2	LA01
3	1100	2100	3/8 Steel tubing	26	26	Inches	2	3	LA01
2	B100	1200	6-Diameter steel plate	1	1	Each	1	4	LA01
2	B100	1300	Hub	1	1	Each	1	5	LA01
2	B100	1400	1/4-20 Screw	4	4	Each	1	6	LA01
1	LA01	S100	Black shade	1	1	Each	0	7	LA01
1	LA01	A100	Socket assembly	1	1	Each	0	8	LA01
2	A100	1500	Steel holder	1	1	Each	8	9	LA01
3	1500	1400	1/4-20 Screw	2	2	Each	9	10	LA01
2	A100	1600	One-way socket	1	1	Each	8	11	LA01
2	A100	1700	Wiring assembly	1	1	Each	8	12	LA01
3	1700	2200	16-Gauge lamp cord	12	12	Feet	12	13	LA01
3	1700	2300	Standard plug terminal	1	1	Each	12	14	LA01

As discussed in the section "Overview: BOM Procedure" on page 15, each indented bill of material can be illustrated by a family tree. In fact, each record in the Indented BOM data set corresponds to a node in this tree. Figure 3.4 displays the tree diagram for the indented bill of material in the IndBOM0 data set. Each tree node or record is uniquely identified by a *sequence* or *ID* number that is assigned to it by the procedure. The Part_ID variable contains this ID number for each record. The Paren_ID variable contains the sequence number for the parent node or parent record. A *parent record* for a given record is the record that defines the parent node of the node defined by the given record. For example, the parent record of record 6 in the data set IndBOM0 is record 1, while record 9 is the parent record of record 10. Record 0 has no parent record.

Figure 3.4 Tree Diagram for the Bill of Material for LA01



The IndBOM0 data set contains other data: The _Level_ variable denotes the indenture level number for each node or record. The root node with the sequence number 0 is at level 0, and the level numbers increase as you look down the tree. The _Prod_ variable contains the part number of the end item. The Qty_Prod variable contains the *quantity per product*, that is, the quantity of the component identified by the _Part_ variable required to make one unit of the end item '`LA01`'. Note that in this particular example, the values of the Qty_Prod variable are identical to the values of the QtyPer variable. This is because the quantities per assembly for all the parent-component relationships are 1, except for relationships '`B100`'-'`1400`', '`1100`'-'`2100`', '`1500`'-'`1400`', '`1700`'-'`2200`'. The components ('`1400`', '`2100`', and '`2200`') of these parent-component relationships do not have any subcomponents themselves.

Figure 3.3, as well as Figure 3.4, shows that the Indented BOM data set lists all the records in *depth-first* order. This scheme lists tree nodes from top to bottom and from left to right. For example, item '`1100`' is listed directly after its parent, '`B100`', and before items '`S100`' and '`A100`' (the right siblings of item '`B100`'). Similarly, item '`2100`' is listed directly after item '`1100`' and before items '`1200`', '`1300`', and '`1400`'. Refer to Aho, Hopcroft, and Ullman (1983) for details about depth-first ordering. See the section "Indented BOM Data Set" on page 32 for details regarding the records in this data set.

Figure 3.5 Summarized Parts List (SumBOM0)

ABC Lamp Company

Summarized Parts List, Period 1

Part	Low_Code	Gros_Req	On_Hand	Net_Req	Desc	Unit
1100	2	1	0	1	Finished shaft	Each
1200	2	1	0	1	6-Diameter steel plate	Each
1300	2	1	0	1	Hub	Each
1400	3	6	0	6	1/4-20 Screw	Each
1500	2	1	0	1	Steel holder	Each
1600	2	1	0	1	One-way socket	Each
1700	2	1	0	1	Wiring assembly	Each
2100	3	26	0	26	3/8 Steel tubing	Inches
2200	3	12	0	12	16-Gauge lamp cord	Feet
2300	3	1	0	1	Standard plug terminal	Each
A100	1	1	0	1	Socket assembly	Each
B100	1	1	0	1	Base assembly	Each
LA01	0	1	0	1	Lamp LA	Each
S100	1	1	0	1	Black shade	Each

The Summarized Parts data set, SumBOM0, is displayed in Figure 3.5. This data set, which has been sorted by the _Part_ variable, has a record for each item in the ABC Lamp Company's product line. Each record contains the part master data from the Part Master data set, PMaster0. The _Part_, Desc, and Unit variables contain the part number, part description, and unit of measure, respectively, for each item. It also contains some other data: The Low_Code variable denotes the *low-level code* of the item. The low-level code for an item is a number that indicates the lowest level in any bill of material at which this item appears. For example, item '1400' appears at levels 2 and 3 in this example; thus, its low-level code is 3. The low-level codes are necessary to make sure that the net requirement for a given item is not calculated until all the gross requirements have been calculated down to that level. The On_Hand variable contains the quantity of the item currently on hand. Since the part master file (as shown in Figure 3.1) does not contain any quantity on hand information, the procedure assumes that this variable is 0 for all items. The Gros_Req and Net_Req variables contain the gross and net requirements, respectively. Again, since the PMaster0 data set does not provide any production plan information, PROC BOM assumes that the final product, 'LA01', is the only master schedule item and the gross requirement for this item is 1 unit. The net requirement for item 'LA01' is the gross requirement (1) minus the quantity on hand (0). The gross and net requirements of the other items are then calculated sequentially:

'B100' Base assembly (1 per lamp)	
Gross requirement	
(= net requirement of 'LA01' × quantity per assembly)	$1 \times 1 = 1$
Quantity on hand	−0
Net requirement	1

'1100' Finished shaft (1 per base assembly)	
Gross requirement	
(= net requirement of 'B100' × quantity per assembly)	$1 \times 1 = 1$
Quantity on hand	−0
Net requirement	1

'2100' 3/8 Steel tubing (26 inches per shaft)	
Gross requirement	
(= net requirement of '1100' × quantity per assembly)	$1 \times 26 = 26$
Quantity on hand	−0
Net requirement	26

If an item (such as item '1400' in this example) is used in more than one assembly, the gross requirement is the total needed in all assemblies where it is used. See the section "Summarized Parts Data Set" on page 37 for details about determining the gross and net requirements.

Recall that the Part Master data set (as shown in Figure 3.1) does not contain any quantity on hand and requirement information, and item 'LA01' is the only final product. Thus, in this example, the summarized parts list (as displayed in Figure 3.5) is actually the same as the summarized bill of material for item 'LA01', with the gross (or net) requirement representing the total usage of each item.

Syntax: BOM Procedure

The following statements are available in PROC BOM.

PROC BOM *options* ;
  STRUCTURE / *options* ;

Functional Summary

The following table outlines the options available for the BOM procedure.

Table 3.1 PROC BOM Options

Description	Statement	Option
Data Set Specification		
Specifies the Product Structure input data set	PROC BOM	DATA=
Specifies the Part Master input data set	PROC BOM	PMDATA=
Specifies the Indented BOM output data set	PROC BOM	OUT=
Specifies the Summarized Parts output data set	PROC BOM	SUMMARYOUT=

Table 3.1 (continued)

Description	Statement	Option
Part Master Information Specification		
Specifies the ID variables	STRUCTURE	ID=
Specifies the lead time variable	STRUCTURE	LEADTIME=
Specifies the part variable	STRUCTURE	PART=
Specifies the quantity on hand variable	STRUCTURE	QTYONHAND=
Specifies the gross requirement variable	STRUCTURE	REQUIREMENT=
Problem Size Options		
Disables use of Utility data set	PROC BOM	NOUTIL
Specifies the number of items	PROC BOM	NPARTS=
Specifies the number of product structure records	PROC BOM	NRELTS=
Product Structure Specification		
Specifies the end items	STRUCTURE	ENDITEM=
Identical Relationships Handling Specification		
Specifies how to handle identical product structure records	PROC BOM	DUPLICATE=
Relationship Information Specification		
Specifies the component variables	STRUCTURE	COMPONENT=
Specifies the scrap factor variables	STRUCTURE	FACTOR=
Specifies the lead-time offset variables	STRUCTURE	OFFSET=
Specifies the parent variable	STRUCTURE	PARENT=
Specifies the quantity per assembly variables	STRUCTURE	QUANTITY=
Specifies the RID variables	STRUCTURE	RID=

PROC BOM Statement

PROC BOM *options* **;**

The PROC BOM statement invokes the procedure. You can specify the following options in the PROC BOM statement.

DATA=*SAS-data-set*

PSDATA=*SAS-data-set*

names the input SAS data set that contains all the product structure records (parent-component relationship, quantity per assembly, scrap factor, and so on.) for a product, product line, plant or company. More than one product structure record can be specified in one observation if they have the same parent item. This data set is also referred to as the Product Structure data set, the Single-Level BOM data set, or the Parent-Component Relationship Structure data set. See the section "Product Structure Data Set" on page 30 for information about the variables that can be included in this data set. If you do not specify the DATA= option, PROC BOM uses the most recently created SAS data set as the input data set.

DUPLICATE=*dup*

specifies how to handle identical product structure records in the Product Structure data set. Two product structure records are called *identical* if they have exactly the same values for the parent, component, lead-time offset, and all the RID variables. The values allowed for this option are

COMBINE

combines the values of the quantity per assembly and scrap factor of all identical product structure records together. See the section "Product Structure Data Set" on page 30 for details about how the procedure combines identical product structure records. The default value is COMBINE.

DISCARD

discards all identical product structure records except the first one.

KEEP

keeps all identical product structure records.

NOUTIL

specifies that the procedure should not use utility data sets for memory management. By default, the procedure resorts to the use of utility data sets and swaps between core memory and utility data sets as necessary if the number of part master or product structure records in the input data sets is larger than the number of part master or product structure records for which memory is initially allocated in core. Specifying this option causes the procedure to increase the memory allocation; if the problem is too large to fit in core memory, PROC BOM stops with an error message.

NPARTS=*nparts*

specifies the number of part master records for which memory is initially allocated in core by the procedure. This enables PROC BOM to make better use of available memory. See the section "Computer Resource Requirements" on page 41 for details. The default value is the number of observations in the Part Master data set.

NRELTS=*nrelts*

NRELATIONSHIPS=*nrelts*

specifies the number of product structure records for which memory is initially allocated in core by the procedure. This enables PROC BOM to make better use of available memory. See the section "Computer Resource Requirements" on page 41 for details. The default value is *nobs* × *ncomp*, where *nobs* is the number of observations in the Product Structure data set and *ncomp* is the number of the Component variables.

OUT=*SAS-data-set*

INDBOM=*SAS-data-set*

specifies a name for the output SAS data set that contains the indented bill of material for all final products in the input data sets or all end items specified in the ENDITEM= option. This data set is also referred to as the Indented BOM data set. See the section "Indented BOM Data Set" on page 32 for information about the variables that are included in this data set. If the OUT= option is omitted, the SAS System creates a data set and names it according to the DATA*n* naming convention.

PMDATA=*SAS-data-set*

PMASTER=*SAS-data-set*

names the input SAS data set that contains all the part (item) master records for a product, product line, plant or company. Each observation in this data set contains one part master record. This data set is also referred to as the Part Master data set. See the section "Part Master Data Set" on page 29 for information about the variables that can be included in this data set. If you do not specify the PMDATA= option, PROC BOM assumes that the part master data are also included in the Product Structure input data set.

SUMMARYOUT=*SAS-data-set*

SUMPART=*SAS-data-set*

specifies a name for the output SAS data set that contains the summarized parts list for the production plan specified in the Part Master data set. This data set is also referred to as the Summarized Parts data set. See the section "Summarized Parts Data Set" on page 37 for information about the variables that are included in this data set. If the SUMMARYOUT= option is omitted, PROC BOM does not produce the summarized parts list.

STRUCTURE Statement

STRUCTURE */ options* **;**

The STRUCTURE statement lists the variables in the Product Structure and the Part Master input data sets. The main variable in the Part Master data set is the Part variable; the main variables in the Product Structure data set are the Parent and Component variables. If all the part master data are contained in the Product Structure data set, you need not specify a Part Master data set.

If two separate input data sets are specified, you must specify both a Part variable and a Parent variable. Otherwise, you may specify either a Parent variable or a Part variable, or both. You must always specify at least one Component variable. The number of Component variables specified must be equal to the number of the product structure records (number of relationships specified) in each observation of the Product Structure data set. All other variables are optional. However, if you specify the Quantity, Offset, or Factor variables, you must have the same number of these variables as the number of Component variables.

In this statement, you can also explicitly specify the items that are to be used as the end (level-0) items.

COMPONENT=*variables*

COMP=*variables*

specifies variables in the Product Structure input data set that contain the part numbers (identifications) of the components that are directly used in the common parent item identified by the Parent variable. These variables are also referred to as Component variables. The Component variables must be of the same type, format, and length as the Part variable in the Part Master data set. At least one Component variable must be specified.

NOTE: This variable is case-sensitive when it is in character format.

ENDITEM=*enditems*

specifies the items that are to be used as the highest level or level-0 items (also referred to as *end items*) in the indented bill of material. In other words, this option can be used to construct indented bill of material and summarized parts list for sub-assemblies. The values for *enditems* must be either

numbers (if the Part variable is numeric) or quoted strings (if the Part variable is character). If you do not specify this option, the procedure assumes every final product (finished good) is an end item and builds an indented bill of material for it.

FACTOR=*variables*

SFACTOR=*variables*

identifies variables in the Product Structure data set that contain the scrap factor information for each product structure record with the parent item identified by the Parent variable and the component identified by each of the Component variables; the *i*th factor variable corresponds to the *i*th Component variable. The *scrap factor* is used to increase the gross requirement to account for anticipated loss within the manufacture of a particular parent item (Cox and Blackstone 1998). These variables are also referred to as the Factor variables. The variables specified must be numeric and the number of Factor variables must be equal to the number of Component variables. If you do not specify this option, PROC BOM assumes the scrap factors for all product structure records to be 0.

ID=*variables*

identifies all variables in the Part Master data set that are not specified in the LEADTIME=, PART=, QTYONHAND=, or REQUIREMENT= options, and are to be included in the Indented BOM and the Summarized Parts output data sets. These variables are also referred to as ID variables. An ID variable cannot be named as: Gros_Req, _Level_, L_Offset, Low_Code, Net_Req, On_Hand, _Parent_, Paren_ID, _Part_, Part_ID, _Prod_, Qty_Per, Qty_Prod, S_Factor, Tot_Lead, or Tot_Off. This option is useful for carrying any identifying and descriptive information (part description, unit of measure, etc.) and control values (lot size, unit cost, etc.) about each item (identified by the Part variable) from the Part Master data set to the output data sets.

LEADTIME=*variable*

DUR=*variable*

identifies the variable in the Part Master data set that contains the lead time information for the item identified by the Part variable. The *lead time* of an item is the length of time between recognition of the need for an order and the receipt of the goods. Individual components of lead time can include order preparation time, queue time, processing time, move or transportation time, and receiving and inspection time (Cox and Blackstone 1998). This variable is also referred to as the LeadTime variable. The variable specified must be numeric. If you do not specify this option, PROC BOM assumes the lead time for each item to be 0. On the other hand, if this option is specified, the procedure creates a new variable, Tot_Lead, in the Indented BOM data set to contain the total lead time. See the section "Indented BOM Data Set" on page 32 for details about the calculation of the total lead time.

OFFSET=*variables*

LTOFFSET=*variables*

identifies variables in the Product Structure data set that contain the lead-time offset information for each product structure record with the parent item identified by the Parent variable and the component identified by each of the Component variables; the *i*th offset variable corresponds to the *i*th Component variable. The *lead-time offset* can be used to control when materials are issued to a work center as well as to determine the need date when planning, purchasing, or manufacturing a particular component (Clement, Coldrick, and Sari 1992). These variables are also referred to as the Offset variables. The variables specified must be numeric and the number of Offset variables must be equal to the number of Component variables. If you do not specify this option, PROC BOM assumes the lead-time offsets for all product structure records to be 0. On the other hand, if this option is specified,

the procedure creates a variable, Tot_Off, in the Indented BOM data set to contain the total offset. See the section "Indented BOM Data Set" on page 32 for details about the calculation of the total offset.

NOTE: The lead-time offset and the total offset can be expressed in units that are different from the lead time.

PARENT=*variable*

specifies the variable in the Product Structure data set that contains the part number or identification for the immediate parent item of the components identified by the Component variables. This variable is also referred to as the Parent variable. The Parent variable must be of the same type, format, and length as the Part variable in the Part Master data set. You must specify this option whenever a Part Master data set is explicitly specified in addition to the Product Structure data set. On the other hand, if a Part Master data set is not specified, the Parent variable is not necessary; if this variable is not specified, the procedure assumes that it is the same as the variable specified in the PART= option.

NOTE: This variable is case-sensitive when it is in character format.

PART=*variable*

ITEM=*variable*

specifies the variable in the Part Master data set that contains the part number or identification. This variable is also referred to as the Part variable. The Part variable can be either character or numeric. You must specify this option whenever a Part Master data set is specified in addition to the Product Structure data set. Otherwise, the Part variable is not necessary; if this variable is not specified, the procedure assumes that it is the same as the variable specified in the PARENT= option.

NOTE: This variable is case-sensitive when it is in character format.

QTYONHAND=*variable*

INVENTORY=*variable*

identifies the variable in the Part Master data set that contains the quantity currently on hand for the item identified by the Part variable. This variable is also referred to as the QtyOnHand variable. The QtyOnHand variable must be numeric. If you do not specify the QtyOnHand variable, the procedure assumes that you do not have any items on hand.

QUANTITY=*variables*

QTYPER=*variables*

identifies variables in the Product Structure data set that contain the quantity per assembly information for each product structure record with the parent item identified by the Parent variable and the components identified by each of the Component variables; the *i*th quantity variable corresponds to the *i*th Component variable. The *quantity per assembly* for a given parent-component relationship is the quantity of the component to be used in the production of 1 unit of the parent item. These variables are also referred to as the Quantity or QtyPer variables. The variables specified must be numeric and the number of Quantity variables must be equal to the number of Component variables. If you do not specify these variables, the procedure assumes that you need 1 unit of each listed component identified by a Component variable to make 1 unit of the parent item identified by the Parent variable.

REQUIREMENT=*variable*

GROSSREQ=*variable*

identifies the variable in the Part Master data set that contains the gross requirement (the quantity planned to be produced in a production plan) for the item identified by the Part variable. This variable

is also referred to as the Requirement variable and must be numeric. If you do not specify this option, the procedure assumes that the gross requirements for all items are missing except for the final products or end items. All final products or end items are assumed to have gross requirements of 1 unit. In other words, if the Part Master data set does not contain the production plan information, PROC BOM uses a default production plan in which each end item has a requirement of 1 unit. The procedure uses the specified (or assumed) values of the Requirement variable to calculate the gross requirements for all other items. These values are saved in the Requirement variable in the Summarized Parts output data set if the SUMMARYOUT= option is specified. If the REQUIREMENT= option is not specified, the Summarized Parts data set uses the default name, Gros_Req.

See the section "Summarized Parts Data Set" on page 37 for details on the calculation of the gross requirements.

RID=*variable*

identifies all variables in the Product Structure data set that are not specified in the COMPONENT=, FACTOR=, OFFSET=, PARENT=, or QUANTITY= options, and are to be included in the Indented BOM data set. These variables are also referred to as RID variables. The RID variables cannot be named as: _Level_, L_Offset, _Parent_, Paren_ID, _Part_, Part_ID, _Prod_, Qty_Per, Qty_Prod, S_Factor, Tot_Lead, or Tot_Off. This option is useful for carrying other information stored in a product structure record to the Indented BOM data set. This typically includes *engineering effectivity*, *operation number*, *reference designator*, and *line sequence number*.

The number of RID variables specified must be a multiple of the number of the Component variables. If you specify m RID and n Component variables as

```
component=(COMP1-COMPn)
RID=(VAR1-VARm)
```

the procedure distributes these m RID variables to each product structure record with the parent item identified by the Parent variable and the component identified by each of the Component variables in the following manner:

component	1**st RID variable**	2**nd RID variable**	...	k**th RID variable**
COMP_1	VAR_1	VAR_{n+1}		VAR_{m-n+1}
COMP_2	VAR_2	VAR_{n+2}		VAR_{m-n+2}
.....				
COMP_n	VAR_n	VAR_{2n}		VAR_m

where $k = m/n$. Note that the procedure treats the variables VAR_1 to VAR_n as the first RID variable for each of the n product structure records, and hence, they must have the same type, format, and length. The same rules apply to variables VAR_{n+1} to VAR_{2n}, ..., VAR_{m-n+1} to VAR_m, etc.

Details: BOM Procedure

Part Master Data Set

The Part Master data set contains all part master records for a product, product line, plant, or company. A typical part master record contains identifying and descriptive data and control values (lead time, lot size, etc.). It may contain data on inventory status (such as quantities on hand), production plan (such as requirements, planned orders, etc.), and costs (Cox and Blackstone 1998). Each observation in this data set represents one part master record. If the data set contains more than one part master record for a given item, only the first one is used.

The BOM procedure uses the Part Master data set as input data with key variable names being used to identify the appropriate information. The Part variable contains the part number or other information that uniquely identifies each item. This variable can be either a numeric or a character variable. The LeadTime, QtyOnHand, and Requirement variables contain the lead time, quantity on hand, and gross requirement, respectively, for the item identified by the Part variable. The QtyOnHand and Requirement variables are only used when the SUMMARYOUT= option is specified. The QtyOnHand variable enables you to include inventory information into the Part Master data set. Similarly, the Requirement variable enables you to specify a production plan in this data set. Other inventory and production plan information that is not directly used by the procedure can also be included in this data set and passed to the output data sets using the ID variables.

The value of the Requirement variable will typically be missing for most items, except for master schedule items. A *master schedule item* (MSI) is an item that is selected to be planned by the master scheduler. In general, final products are master schedule items. Some companies like to select a few items that are deemed critical in their impact on lower-level components or resources as master schedule items. The requirements of the master schedule items are also called *independent demands*. In the Part Master data set, if the value of the Requirement variable is greater than or equal to 0, the procedure treats the item identified by the Part variable as an MSI and assumes that its gross requirement is planned by the master scheduler. The gross requirements of other items (also known as the *dependent demands*) are determined by the procedure as the value of the net requirement of each parent item times the quantity required to make one unit of the parent item, summed over all the parent items. This process is called the *dependent demand process* (Clement, Coldrick, and Sari 1992). See the section "Summarized Parts Data Set" on page 37 for details about determining gross and net requirements.

Other part master data such as part description, lot size, order quantity, unit of measure, unit of cost, or planner/buyer code can be passed to the Indented BOM and Summarized Parts output data sets using the ID variables.

Table 3.2 lists all the variables in this input data set, with their type and their interpretation by the BOM procedure. It also lists the options in the STRUCTURE statement that are used to identify these variables.

Table 3.2 Part Master Data Set and Associated Variables

Variable	Type	Option	Interpretation
ID	Character or numeric	ID=	Additional master data for the item
LeadTime	Numeric	LEADTIME=	Lead time of the item

Table 3.2 (continued)

Variable	Type	Option	Interpretation
Part	Character or numeric	PART=	Part number or identification for the item
QtyOnHand	Numeric	QTYONHAND=	Quantity of the item that is currently on hand
Requirement	Numeric	REQUIREMENT=	Gross requirement of the item

You can combine the Part Master data set and the Product Structure data set together to create a new data set and use it as the input data set for PROC BOM. See the SIBOM1 data set shown in Output 3.1.1 and the SIBOM2 data set shown in Output 3.2.1 for examples.

Product Structure Data Set

The Product Structure data set lists all product structure records in a product, product line, plant, or company. Each product structure record defines the relationship of one component to its immediate parent item. It also includes quantity per assembly. The *quantity per assembly* is the quantity of the component that is required to make one unit of the parent item. A product structure record may contain fields for scrap factor, lead-time offset, and other information. The *scrap factor* of a parent-component relationship is used to increase the gross requirement of the component to account for anticipated loss within the manufacture of the parent item (Cox and Blackstone 1998). The *lead-time offset* of a parent-component relationship can be used to control when the component is issued to a work center while manufacturing the parent item, as well as to determine the need date when planning, purchasing, or manufacturing the component (Clement, Coldrick, and Sari 1992). Note that the lead-time offset may be expressed in a unit that is different from the lead time in the Part Master data set. For example, the lead time may be measured in weeks while the lead-time offset is measured in days or hours. If necessary, you can pass the unit that is used to measure the lead-time offset from this data set to the Indented BOM data set using the RID= option. Similarly, you can pass the unit for expressing the lead time to the Indented BOM and Summarized Parts data sets from the Part Master data set using the ID= option.

You can put more than one product structure record with the same parent item into one observation in this data set. See the SIBOM1 data set shown in Output 3.1.1 as an example. You can also include part master data in this data set. See Example 3.1 and Example 3.2 as examples.

The BOM procedure uses the Product Structure data set as input data with key variable names being used to identify the appropriate information. The Parent and Component variables contain the part numbers for the parent item and its components. These variables must be of the same type, format, and length as the Part variable in the Part Master data set. The Quantity, Factor, and Offset variables contain information for the quantity per assembly, scrap factor, and lead-time offset, respectively. Additional information for the parent-component relationships, such as engineering effectivity, point-of-use, operation number, reference designator, line sequence number, etc., can be passed to the Indented BOM data set using the RID variables.

Table 3.3 lists all the variables in this input data set, with their type and their interpretation by the BOM procedure. It also lists the options in the STRUCTURE statement that are used to identify these variables.

Table 3.3 Product Structure Data Set and Associated Variables

Variable	Type	Option	Interpretation
Component	Same as Part variable	COMPONENT=	Part number for the component
Factor	Numeric	FACTOR=	Scrap factor for the relationship
Offset	Numeric	OFFSET=	Lead-time offset for the relationship
Parent	Same as Part variable	PARENT=	Part number for the parent item
Quantity	Numeric	QUANTITY=	Quantity per assembly for the relationship
RID	Character or numeric	RID=	Additional information about the relationship

A given component can be structured into a parent item more than once. The Offset variable (if the component is used at a different time in the process) and any RID variables (point-of-use, parent operation, line sequence number, etc.) can be used to distinguish product structure records with the same parent and component. See Example 3.6 as an example. If two or more product structure records have the same parent, component, lead-time offset, and values for all RID variables, the procedure handles them according to the specification of the DUPLICATE= option. If the value of the DUPLICATE= option is specified as the keyword *COMBINE*, the procedure combines these product structure records into one record. The quantity per assembly and the scrap factor of this new product structure record are determined as

$$q_0 = \sum_{i=1}^{k} q_i$$

and

$$f_0 = \frac{\sum_{i=1}^{k} (q_i \times f_i)}{q_0}$$

respectively, where

k = number of identical product structure records
f_0 = value of the scrap factor of the new product structure record
f_i = value of the scrap factor of the ith identical product structure record
q_0 = value of the quantity per assembly of the new product structure record
q_i = value of the quantity per assembly of the ith identical product structure record

If the value of the DUPLICATE= option is specified as the keyword *DISCARD*, then the procedure keeps the first identical product structure record and discards the others. On the other hand, if the value of the DUPLICATE= option is specified as the keyword *KEEP*, PROC BOM keeps all identical product structure records and processes them independently.

Indented BOM Data Set

The Indented BOM data set produced by PROC BOM contains the indented bill of material for all final products in the Part Master data set, or for all end items specified in the ENDITEM= option. Each observation of this data set represents one indented BOM record. Information contained in each indented BOM record can be classified into three categories: part master data for the item identified by the _Part_ variable, product structure data for the relationship of the component identified by the _Part_ variable to its parent identified by the _Parent_ variable, and data that are dedicated to this record.

As discussed in the section "Getting Started: BOM Procedure" on page 17, each indented bill of material can be illustrated by a family tree. Each node in the family tree corresponds to a record in the Indented BOM data set. The indented BOM record that is associated with the root node of a family tree is distinguished as the *root record* of the indented bill of material. For a given indented BOM record and its corresponding node j, the *parent record* of record j is the indented BOM record that is associated with the parent node of node j. If the indented BOM record i is the parent record of record j, then record j is a *child record* of record i. Every record in the Indented BOM data set has a parent record except for the root records. Alternatively, an indented BOM record can have zero, one, two, or more child records.

Like the product structure record in the Product Structure data set, each indented BOM record in the Indented BOM data set contains a relationship of one component to its immediate parent item. The _Part_ variable contains the part number or identification information of the component. The _Parent_ variable contains the part number for the immediate parent item. Each record also contains all other product structure data from the product structure record with the same parent item (identified by the Parent variable) and component (identified by the Component variable) as in this record. The BOM procedure uses the following convention for naming the variable identifying the product structure data.

If you specify exactly one Quantity variable, the procedure uses the same variable name as specified in the QUANTITY= option for the variable that contains the value of the quantity per assembly for the relationship. On the other hand, if you do not specify the QUANTITY= option, or if you specify more than one Quantity variable, the variable Qty_Per contains this value. If you do not specify the QUANTITY= option, the value of the quantity per assembly is 1 for all relationships. The procedure also uses the same variable name as specified in the FACTOR= option if only one Factor variable is specified, and uses the default name, S_Factor, if two or more Factor variables are specified, for the variable that contains the value of scrap factor. Similarly, if you specify more than one Offset variable, the default name, L_Offset, is used; if you specify only one Offset variable, the same variable name is used for the variable that contains the value of the lead-time offset. Note that if you do not specify the FACTOR= or OFFSET= options, the Indented BOM data set does not have variables that contain the values of scrap factor or lead-time offset, respectively. The Indented BOM data set has the same RID variables as in the Product Structure data set. The values of those variables for each record are copied from the product structure record with the same parent item and component. The values of the _Parent_, Factor, Offset, Quantity, and RID variables for any root records should be missing, if these variables are present.

Each record of the Indented BOM data set also contains part master data for the item identified by the _Part_ variable in this indented BOM record. For instance, the lead time information is contained in the variable

with the same variable name as specified in the LEADTIME= option. The Indented BOM data set also has the same ID variables as specified in the ID= option. The values of these variables for the indented BOM record are copied from the corresponding variables in the part master record for the same item (identified by the _Part_ variable) in the Part Master data set.

In addition to the product structure and part master data, each record of the Indented BOM data set contains other data calculated by PROC BOM. The _Prod_ variable contains the part number for the final product or end item of the indented bill of material corresponding to this record. For any root records, the value of the _Prod_ variable is the same as the value of the _Part_ variable. The _Level_ variable contains the indenture level number of the record. The procedure assigns the value of the indenture level to each record as follows: Each root record has indenture level 0, and all child records corresponding to a root record have level 1. All subsequent child records of these records have the level number increased by 1. This process continues until there are no further child records. The Part_ID variable contains the sequence or ID number that uniquely identifies each record in this data set. PROC BOM assigns this sequence number to each record in the data set by using the *depth-first numbering* scheme (Aho, Hopcroft, and Ullman 1983). The Paren_ID variable contains the ID number for the immediate parent record. The value of the Paren_ID variable for a root record is missing. These two variables are useful when you use the Indented BOM data set as the input data set for other SAS/OR procedures, such as PROC NETDRAW and PROC CPM, which require unique identification of each node. Refer to "The NETDRAW Procedure" and "The CPM Procedure" chapters in the *SAS/OR User's Guide: Project Management* for details.

The Qty_Prod variable denotes the quantity of the item (identified by the _Part_ variable) required to make one unit of the end item identified by the _Prod_ variable. This value is also known as *quantity per product* and can be determined as

$$qp_i = \begin{cases} 1 & \text{if } j \text{ is missing} \\ qp_j \times q_i & \text{otherwise} \end{cases}$$

where

$$\begin{aligned} i &= \text{the value for the Part\_ID variable} \\ j &= \text{the value for the Paren\_ID variable} \\ qp_i &= \text{the quantity per product of record } i \\ q_i &= \text{the quantity per assembly of record } i \end{aligned}$$

If the LEADTIME= option is specified, PROC BOM measures the total lead time accumulated from the root record to the current record and stores it in the Tot_Lead variable. The formula to determine the total lead time for each record is

$$tl_i = \begin{cases} t_i & \text{if } j \text{ is missing} \\ tl_j + t_i & \text{otherwise} \end{cases}$$

where

$$
\begin{aligned}
i &= \text{the value for the Part\_ID variable} \\
j &= \text{the value for the Paren\_ID variable} \\
t_i &= \text{the lead time of record } i \\
tl_i &= \text{the total lead time of record } i
\end{aligned}
$$

Similarly, if the OFFSET= option is specified, the procedure computes the total offset from the root record to the current record and stores it in the Tot_Off variable. The total offset for each record can be determined as

$$
to_i = \begin{cases} o_i & \text{if } j \text{ is missing} \\ to_j + o_i & \text{otherwise} \end{cases}
$$

where

$$
\begin{aligned}
i &= \text{the value for the Part\_ID variable} \\
j &= \text{the value for the Paren\_ID variable} \\
o_i &= \text{the lead-time offset of record } i \\
to_i &= \text{the total offset of record } i
\end{aligned}
$$

Note that the lead-time offset and the total offset may be expressed in a unit that is different from the lead time and the total lead time in this data set.

Table 3.4 lists all the variables in the Indented BOM data set. It also lists the type and a brief description of these variables.

Table 3.4 Indented BOM Data Set and Associated Variables

Product Structure Data (for the parent-component relationship)		
Variable	**Type**	**Interpretation**
Factor	Numeric	Scrap factor
Offset	Numeric	Lead-time offset
Parent	Same as _Part_	Part number for the parent item
Part	Character or numeric	Part number for the component
Quantity	Numeric	Quantity per assembly
RID	Character or numeric	Additional product structure data
Part Master Data (for the component)		
Variable	**Type**	**Interpretation**
ID	Character or numeric	Additional part master data
LeadTime	Numeric	Lead time

Table 3.4 (continued)

Indented BOM Data (for the record)		
Variable	**Type**	**Interpretation**
Level	Numeric	Indenture level number
Paren_ID	Numeric	ID number of the parent record
Part_ID	Numeric	ID number
Prod	Same as _Part_	Part number for the end item
Qty_Prod	Numeric	Quantity per product
Tot_Lead	Numeric	Total lead time
Tot_Off	Numeric	Total lead-time offset

The indented BOM records in this data set are organized so that the bill of material contained in the data set are listed one by one. In each bill of material, the root record is always listed first. In addition, the left-most (oldest) component of each parent item is listed directly after its parent and before any right siblings of the parent item. For example, from the Indented BOM data shown in Figure 3.3, you can easily see that the final product '`LA01`' is the first record of the Indented BOM data set. Moreover, item '`1100`' is listed directly after its parent, '`B100`', and before the items '`S100`' and '`A100`' (the right siblings of the item '`B100`'). The item '`1100`' is followed immediately by its first component '`2100`' (the only component in this case), and so on.

By organizing indented BOM records in such a manner and using the indenture level information, the single-level and multilevel relationships are quite clear. Let us use the indented BOM data set displayed in Figure 3.3 as an example. The three components, '`B100`', '`S100`', and '`A100`', that go into '`LA01`' are easily determined by the level 1 identifiers. It is also easy to see that item '`A100`' takes three components at level 2, and one of them, item '`1500`', has a level 3 component (item '`1400`') and another item, '`1700`', has two level 3 components (items '`2200`' and '`2300`'). In fact, record 1 to record 6 of this data set contain the indented bill of material for item '`B100`' because record 7 is the first record to have the same level as record 1. The *single-level where-used* and *indented where-used* lists can also be done in the same manner but by retrieving records in reversed order. For example, items '`B100`' and '`1500`' that directly use component '`1400`' can be detected either by looking at the values of the _Parent_ variable directly in record 6 and record 10, or by looking up at the first records with level numbers that are 1 less than record 6 and record 10, respectively. By continuing to look upward, you see that item '`1500`' is used in '`A100`' and item '`A100`' is used in '`LA01`'. As discussed previously, you can also find the parent record (the first record with one level up while retrieving the data in reversed order) information in the Paren_ID variable. The Indented BOM data set can also be used to propagate gross requirements, determine a schedule of lower-level items needed, aggregate lower-level demands, roll up material costs, and so on. See the section "Bill of Material Explosion and Implosion" on page 35 for details.

Bill of Material Explosion and Implosion

As described previously, the records (or observations) in the Indented BOM data set are organized in depth-first order (Aho, Hopcroft, and Ullman 1983). The order of observations in the Indented BOM data set and the presence of the _Level_ variable make it possible to perform bill of material explosion and implosion

easily. In fact, the Indented BOM data set contains the quantity per product, total lead time, and total lead-time offset data already. The Summarized Parts data set lists the gross and net requirements for each item to fill a given production plan. Other calculations can be performed using a simple SAS DATA step. For example, the following SAS code creates a single-level bill of material for the item '`LA01`' from the Indented BOM data set, IndBOM0, as displayed in Figure 3.3. The single-level bill of material is shown in Figure 3.6.

```
  /* Display the components that are directly used */
  /* in item LA01                                  */
proc print data=IndBOM0(where=(_Parent_='LA01')
                        rename=(_Part_=Component))
          noobs;
   var Component Desc QtyPer Unit;
   title 'ABC Lamp Company';
   title3 'Single-level Bill of Material Retrieval, Part LA01';
run;
```

Figure 3.6 Components Directly Used in Part LA01

ABC Lamp Company

Single-level Bill of Material Retrieval, Part LA01

Component	Desc	Qty Per	Unit
B100	Base assembly	1	Each
S100	Black shade	1	Each
A100	Socket assembly	1	Each

The single-level where-used list for item '`1400`', displayed in Figure 3.7, is created with the following SAS code. PROC SQL is invoked to link the part master data to each parent item.

```
  /* Create the where-used data set */
data Used0a(keep=_Parent_ Paren_ID QtyPer Unit);
   set IndBOM0(where=(_Part_='1400'));
run;

  /* Get the part description from the IndBOM0 data set */
proc sql;
   create table Used0b as
      select Used0a._Parent_, IndBOM0.Desc,
             Used0a.QtyPer, Used0a.Unit
         from Used0a left join IndBOM0
            on Used0a.Paren_ID=IndBOM0.Part_ID;
quit;

  /* Display the where-used data set */
proc print data=Used0b noobs;
   var _Parent_ Desc QtyPer Unit;
   title 'ABC Lamp Company';
   title3 'Single-level Where-used Report, Part 1400';
run;
```

Figure 3.7 Parents in Which Part 1400 is Directly Used

ABC Lamp Company

Single-level Where-used Report, Part 1400

Parent	Desc	Qty Per	Unit
B100	Base assembly	4	Each
1500	Steel holder	2	Each

The section "Reporting Macros" on page 86 in Chapter 4, "Bill of Material Postprocessing Macros," describes six SAS autocall macros that can be used to create single-level, indented, and summarized bill of material and where-used lists for an item from the Indented BOM data set. Example 3.8 demonstrates a simple way to roll up costs using SAS DATA step code. Similar techniques can be used to aggregate lower level requirements up to the end items or to perform other bill of material explosion and implosion. See Example 3.9 and Example 3.10 for further examples.

Summarized Parts Data Set

If the SUMMARYOUT= option in the PROC BOM statement is specified, the BOM procedure creates a Summarized Parts data set. This data set lists all the items with their quantities required (needed to be made or ordered) in order to fill the production plan specified in the Part Master data set, taking into account the quantities of these items on hand in the planning period. This data set lists each item once for the total quantity needed. Each observation in this data set represents one record, which is uniquely identified by the part number in the _Part_ variable. Each record contains all part master data (from the Part Master data set) for the item identified by the _Part_ variable. Each record also contains fields for net requirement and low-level code.

The _Part_ variable contains the part number or identification information of the item. The data set has the same LeadTime, QtyOnHand, Requirement, and all ID variables as in the Part Master data set. If you do not specify the QTYONHAND= option, the default name, On_Hand, is used to name the variable that contains the quantity of the item identified by the _Part_ variable that is currently on hand in the planning period. The default value of this variable is 0, since the procedure assumes there are no items on hand if the QTYONHAND= option is not specified. Similarly, if the REQUIREMENT= option is not specified, the default name Gros_Req is used to name the variable that contains the value of the gross requirement for the item identified by the _Part_ variable. The values for these variables, except for the variable that contains the gross requirement, are carried from the Part Master data set. The gross requirements for all items, except for master schedule items, are determined by the dependent demand process, which is described later.

The Net_Req variable contains the value of the net requirement and the Low_Code variable contains the value of the low-level code for the item identified by the _Part_ variable. The low-level code of an item is the lowest indenture level in any bill of material containing the item. The low-level codes are necessary to make sure that the net requirement of any item is not calculated until all the gross requirements have been calculated down to that level.

The gross and net requirements for all items are determined in the order of increasing low-level codes. Start at an item with the smallest low-level code (usually 0). If an item c is a master schedule item, the gross requirement for this item is carried from the Part Master data set. Otherwise, it is given by

$$R_c = \sum_{\text{all}(p,c)\text{in}\Re} [N_p \times q_{(p,c)} \times (1 + e_{(p,c)})]$$

and the net requirement is determined as

$$N_c = \begin{cases} R_c - H_c & \text{if } R_c > H_c \\ 0 & \text{otherwise} \end{cases}$$

where

(p, c) = a relationship between the parent item p and the component c
$\Re$ = a collection of all parent-component relationships that are defined in the Indented BOM data set
R_c = the gross requirement for the item c
N_c = the net requirement for the item c
H_c = the quantity on hand for the item c
$q_{(p,c)}$ = the quantity per assembly for the relationship (p, c)
$e_{(p,c)}$ = the scrap factor for the relationship (p, c)

This computational process continues for items with larger low-level codes until the values of the gross and net requirements for all components in the Indented BOM data set have been determined. This computational process is known as the *dependent demand process*.

A summarized parts list should not be confused with a summarized bill of material. As discussed previously, a summarized parts list shows all the items and their quantities required to fill a pre-specified production plan. The gross and net requirements in this list are calculated taking into account the quantity of each item that is available in the planning period. On the other hand, a summarized bill of material for a pre-specified item lists all the components and their total quantities used in making one batch of that item, without any regard for quantities of items on hand. In a special case when the input data do not contain any requirement, quantity on hand, or scrap factor information, and the Indented BOM data set contains only one end item, then the summarized parts list in this data set happens to be the same as the summarized bill of material for the end item.

Table 3.5 lists all the variables in the Summarized Parts data set. It also lists the type and a brief description of these variables.

Table 3.5 Summarized Parts Data Set and Associated Variables

Variable	Type	Interpretation
ID	Character or numeric	Additional master data for the item
LeadTime	Numeric	Lead time for the item
Low_Code	Numeric	Low-level code for the item
Net_Req	Numeric	Net requirement for the item
Part	Character or numeric	Part number for the item
QtyOnHand	Numeric	Quantity of the item that is currently on hand
Requirement	Numeric	Gross requirement for the item

Missing Values in the Input Data Sets

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

Table 3.6 Treatment of Missing Values in the BOM Procedure

Data Set	Variable	Value Used/Assumption Made/Action Taken
Part Master	ID	Missing, if Part is not missing; otherwise ignored
	LeadTime	0, if Part is not missing; otherwise ignored
	Part	Value ignored
	QtyOnHand	0, if Part is not missing; otherwise ignored
	Requirement	Ignored if Part is missing; 1, if Part is not missing and the item identified by the Part variable is an end item; otherwise, the value is determined by the procedure
Product Structure	Component	Value ignored
	Factor	0, if corresponding Component variable is not missing; otherwise ignored
	Offset	0, if corresponding Component variable is not missing; otherwise ignored

Table 3.6 (continued)

Data Set	Variable	Value Used/Assumption Made/Action Taken
	Parent	Input error: procedure stops with error message, if this is the first observation; otherwise, the value of the Parent variable in the previous observation
	Quantity	1, if corresponding Component variable is not missing; otherwise ignored
	RID	Missing, if corresponding Component variable is not missing; otherwise ignored

Note that in the Product Structure data set, a missing value is allowed for the Parent variable only when two or more consecutive observations contain product structure records with the same parent item (see Output 3.1.1 and Output 3.3.2 as examples). In the Part Master data set, if the Part variable is missing, the values for the ID, LeadTime, QtyOnHand, and Requirement variables are ignored by the procedure. If the Part variable is not missing but the Requirement variable value is missing, the gross requirement of the item identified by the Part variable is assumed to be 1 if the item in question is an end item. Otherwise, the gross requirement of this item is determined based on the dependent demand process. See the section "Summarized Parts Data Set" on page 37 for details about the dependent demand process.

Macro Variable _ORBOM_

The BOM procedure defines a macro variable named _ORBOM_. This variable contains a character string that indicates the status of the procedure. It is set at procedure termination. The form of the _ORBOM_ character string is STATUS= REASON=, where STATUS= is either SUCCESSFUL or ERROR_EXIT, and REASON= (if PROC BOM terminated unsuccessfully) can be one of the following:

CYCLE

BADDATA_ERROR

MEMORY_ERROR

IO_ERROR

SEMANTIC_ERROR

SYNTAX_ERROR

BOM_BUG

UNKNOWN_ERROR

This information can be used when PROC BOM is one step in a larger program that needs to determine whether the procedure terminated successfully. Because _ORBOM_ is a standard SAS macro variable, it can be used in the ways that all macro variables can be used.

Computer Resource Requirements

There is no inherent limit on the size of the problem that can be handled with the BOM procedure. The number of items and relationships are constrained only by the amount of memory available. Naturally, there needs to be a sufficient amount of core memory available in order to invoke and initialize the SAS System. As far as possible, the procedure attempts to store all the data in core memory.

However, if the problem is too large to fit in core memory, the procedure resorts to the use of utility data sets and swaps between core memory and utility data sets as necessary, unless the NOUTIL option is specified. The procedure uses the NPARTS= and NRELTS= 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 Product Structure and Part Master data sets. See the section "PROC BOM Statement" on page 23 for default specifications.

The storage requirement for the data area and the time required by the procedure are proportional to the numbers of items and parent-component relationships in the problem.

Examples: BOM Procedure

This section contains examples that illustrate the features of the BOM procedure. Most of the examples use the data from the ABC Lamp Company product line described in the section "Getting Started: BOM Procedure" on page 17.

Example 3.1: Bill of Material with Single Input Data Set

This example uses a single input data set for invoking the BOM procedure. It also demonstrates the use of PROC NETDRAW to draw a tree diagram for the indented bill of material contained in the Indented BOM output data set. The input data set, SIBOM1, is depicted in Output 3.1.1. Each observation of this data set contains up to 3 product structure records. It also contains part master data, namely, the part description denoted by the Desc variable and the unit of measure denoted by the Unit variable, for the item identified by the value of the Parent variable. Note that the parent item '`B100`' has 4 components, causing the specification to require two observations. The values of the Parent, Desc, and Unit variables for the second observation are missing because they have the same values as the previous observation.

Output 3.1.1 The Input Data Set (SlBOM1)

ABC Lamp Company

PROC BOM Input Data Set

Parent	Desc	Unit	Comp1	Comp2	Comp3	Qty1	Qty2	Qty3
LA01	Lamp LA	Each	B100	S100	A100	1	1	1
B100	Base assembly	Each	1100	1200	1300	1	1	1
			1400			4	.	.
S100	Black shade	Each				.	.	.
A100	Socket assembly	Each	1500	1600	1700	1	1	1
1100	Finished shaft	Each	2100			26	.	.
1200	6-Diameter steel plate	Each				.	.	.
1300	Hub	Each				.	.	.
1400	1/4-20 Screw	Each				.	.	.
1500	Steel holder	Each	1400			2	.	.
1600	One-way socket	Each				.	.	.
1700	Wiring assembly	Each	2200	2300		12	1	.
2100	3/8 Steel tubing	Inches				.	.	.
2200	16-Gauge lamp cord	Feet				.	.	.
2300	Standard plug terminal	Each				.	.	.

The following code invokes PROC BOM to produce the indented bill of material and the summarized parts list. The indented bill of material is displayed in Output 3.1.2, and the summarized parts list, which has been sorted by the _Part_ variable, is displayed in Output 3.1.3. These are exactly the same as the indented bill of material shown in Figure 3.3 and the summarized parts list shown in Figure 3.5.

```
/* Create the indented BOM and the summarized parts list */
proc bom data=SlBOM1 out=IndBOM1 summaryout=SumBOM1;
   structure / part=Parent
               parent=Parent
               component=(Comp1-Comp3)
               quantity=(Qty1-Qty3)
               id=(Desc Unit);
run;
```

Output 3.1.2 Indented Bill of Material (IndBOM1)

ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Desc	Qty_Per	Qty_Prod	Unit	Parent_ID	Part_ID	_Prod_
0		LA01	Lamp LA	.	1	Each	.	0	LA01
1	LA01	B100	Base assembly	1	1	Each	0	1	LA01
2	B100	1100	Finished shaft	1	1	Each	1	2	LA01
3	1100	2100	3/8 Steel tubing	26	26	Inches	2	3	LA01
2	B100	1200	6-Diameter steel plate	1	1	Each	1	4	LA01
2	B100	1300	Hub	1	1	Each	1	5	LA01
2	B100	1400	1/4-20 Screw	4	4	Each	1	6	LA01
1	LA01	S100	Black shade	1	1	Each	0	7	LA01
1	LA01	A100	Socket assembly	1	1	Each	0	8	LA01
2	A100	1500	Steel holder	1	1	Each	8	9	LA01
3	1500	1400	1/4-20 Screw	2	2	Each	9	10	LA01
2	A100	1600	One-way socket	1	1	Each	8	11	LA01
2	A100	1700	Wiring assembly	1	1	Each	8	12	LA01
3	1700	2200	16-Gauge lamp cord	12	12	Feet	12	13	LA01
3	1700	2300	Standard plug terminal	1	1	Each	12	14	LA01

Output 3.1.3 Summarized Parts List (SumBOM1)

ABC Lamp Company

Summarized Parts List, Period 1

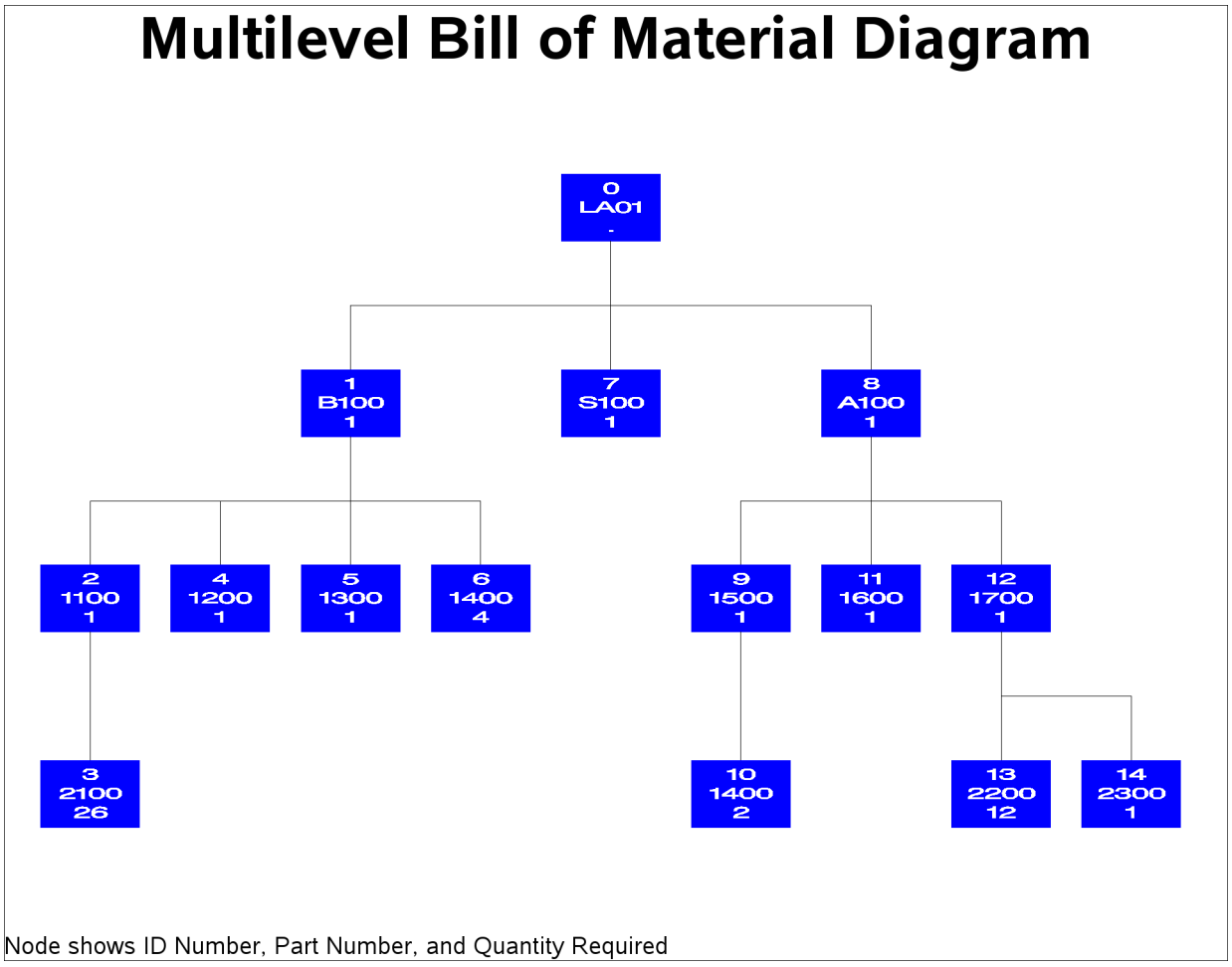
Part	Low_Code	Gros_Req	On_Hand	Net_Req	Desc	Unit
1100	2	1	0	1	Finished shaft	Each
1200	2	1	0	1	6-Diameter steel plate	Each
1300	2	1	0	1	Hub	Each
1400	3	6	0	6	1/4-20 Screw	Each
1500	2	1	0	1	Steel holder	Each
1600	2	1	0	1	One-way socket	Each
1700	2	1	0	1	Wiring assembly	Each
2100	3	26	0	26	3/8 Steel tubing	Inches
2200	3	12	0	12	16-Gauge lamp cord	Feet
2300	3	1	0	1	Standard plug terminal	Each
A100	1	1	0	1	Socket assembly	Each
B100	1	1	0	1	Base assembly	Each
LA01	0	1	0	1	Lamp LA	Each
S100	1	1	0	1	Black shade	Each

As discussed in the section "Summarized Parts Data Set" on page 37, since item 'LA01' is the only master schedule item in this example, and both the quantity on hand and the scrap factor are zero for all items, the summarized parts list displayed in Output 3.1.3 is also the summarized bill of material for item 'LA01'.

The following SAS code uses PROC NETDRAW to draw a *family tree* diagram displaying the indented bill of material for item 'LA01'. The NETDRAW procedure requires the descriptive information for each node to be associated with the parent node in each observation. However, the Indented BOM data set contains the descriptive (part master) information for the component identified by the _Part_ variable. Thus, the following code prepares the data for PROC NETDRAW. This code first creates two data sets from the Indented BOM data set, IndBOM1. The IndBOM1a data set contains all variables in the Indented BOM data set, except for the Part_ID variable. The Part_ID variable is dropped and its value is copied to the Paren_ID variable. Each record in this data set is represented by a tree node in the diagram, which is uniquely identified by the value of the Paren_ID variable. The other data set, IndBOM1b, contains all parent-component relationships, in which each relationship is identified by the Paren_ID and Part_ID variables. Each observation in this data set is represented by a link in the diagram. These two data sets are concatenated to form the input data set for PROC NETDRAW. The tree diagram for the indented bill of material for item 'LA01' is displayed in Output 3.1.4. For further details about the NETDRAW procedure, refer to Chapter 9, "The NETDRAW Procedure" (*SAS/OR User's Guide: Project Management*).

```
/* Draw a tree diagram for illustrating the product structure */
/* Each record denotes a node in the tree */
data IndBOM1a(drop=Part_ID);
   set IndBOM1;
   Paren_ID=Part_ID;
run;

/* Extract the Parent - Part information */
data IndBOM1b;
   set IndBOM1(keep=Paren_ID Part_ID);
run;

/* Prepare the data set for running NETDRAW */
data TreBOM1;
   set IndBOM1a IndBOM1b;
run;

/* Specify graphics options */
goptions hpos=32 vpos=80 border;
pattern1 v=s c=blue;

title h=5 j=c 'Multilevel Bill of Material Diagram';
footnote h=2 j=l
   'Node shows ID Number, Part Number, and Quantity Required';


/* Invoke PROC NETDRAW to display BOM tree */
proc netdraw data=TreBOM1( where=(Paren_ID NE .) ) out=NetOUT;
   actnet / act=Paren_ID succ=Part_ID id=(Paren_ID _Part_ Qty_Per)
            ctext=white htext=3 font=swiss carcs=black
            ybetween=3 xbetween=8 centerid
            tree pcompress rotatetext rotate
            arrowhead=0 rectilinear nodefid nolabel;
run;
```

Output 3.1.4 Tree Diagram for the Bill of Material



Example 3.2: Bill of Material with Lead Time Information

As in Example 3.1, this example also uses a single input data set with each observation containing product structure data and part master data. Unlike the SIBOM1 data set (shown in Output 3.1.1) in which each observation contains three product structure records, each observation in the input data set SIBOM2, as depicted in Output 3.2.1, contains only one product structure record. The observations in both input data sets also contain one part master record. However, in the previous example, the variable Parent contains the part number for the part master record, whereas in the current example, the part number for the part master record is contained in the variable Component. In addition, in this example, each part master record also contains requirement (contained in the Gros_Req variable) and quantity on hand (contained in the On_Hand variable) information.

Output 3.2.1 The Input Data Set (SlBOM2)

ABC Lamp Company

PROC BOM Input Data Set

Parent	Component	Desc	Unit	Lead Time	Qty Per	Gros_Req	On_Hand
	LA01	Lamp LA	Each	2	.	50	20
LA01	B100	Base assembly	Each	1	1	.	50
LA01	S100	Black shade	Each	2	1	.	.
LA01	A100	Socket assembly	Each	1	1	.	.
B100	1100	Finished shaft	Each	2	1	.	.
B100	1200	6-Diameter steel plate	Each	3	1	.	.
B100	1300	Hub	Each	2	1	.	.
B100	1400	1/4-20 Screw	Each	1	4	.	.
A100	1500	Steel holder	Each	2	1	.	.
A100	1600	One-way socket	Each	2	1	.	.
A100	1700	Wiring assembly	Each	1	1	.	.
1100	2100	3/8 Steel tubing	Inches	3	26	.	.
1500	1400	1/4-20 Screw	Each	1	2	.	.
1700	2200	16-Gauge lamp cord	Feet	2	12	.	.
1700	2300	Standard plug terminal	Each	1	1	.	.

The following code produces the Indented BOM data set and the Summarized Parts data set. The Indented BOM data set, IndBOM2, is displayed in Output 3.2.2. The "PART=Component" specification in the STRUCTURE statement indicates that the Component variable contains the part number for the part master record in each observation. Since the LEADTIME= option is specified, a new variable, Tot_Lead, has been added to the Indented BOM data set. This variable denotes the total lead time accumulated from the root record (the top-most record) to the record identified by the value of the Part_ID variable.

```
  /* Create the indented BOM and the summarized parts list */
proc bom data=SlBOM2 out=IndBOM2 summaryout=SumBOM2;
   structure / part=Component
               leadtime=LeadTime
               parent=Parent
               component=Component
               quantity=QtyPer
               qtyonhand=On_hand
               requirement=Gros_Req
               id=(Desc Unit);
run;
```

Output 3.2.2 Indented Bill of Material with Lead Time (IndBOM2)

ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Desc	Qty_Per	Qty_Prod	Unit	Lead_Time	Tot_Lead_Time	_Prod_
0		LA01	Lamp LA	.	1	Each	2	2	LA01
1	LA01	B100	Base assembly	1	1	Each	1	3	LA01
2	B100	1100	Finished shaft	1	1	Each	2	5	LA01
3	1100	2100	3/8 Steel tubing	26	26	Inches	3	8	LA01
2	B100	1200	6-Diameter steel plate	1	1	Each	3	6	LA01
2	B100	1300	Hub	1	1	Each	2	5	LA01
2	B100	1400	1/4-20 Screw	4	4	Each	1	4	LA01
1	LA01	S100	Black shade	1	1	Each	2	4	LA01
1	LA01	A100	Socket assembly	1	1	Each	1	3	LA01
2	A100	1500	Steel holder	1	1	Each	2	5	LA01
3	1500	1400	1/4-20 Screw	2	2	Each	1	6	LA01
2	A100	1600	One-way socket	1	1	Each	2	5	LA01
2	A100	1700	Wiring assembly	1	1	Each	1	4	LA01
3	1700	2200	16-Gauge lamp cord	12	12	Feet	2	6	LA01
3	1700	2300	Standard plug terminal	1	1	Each	1	5	LA01

Output 3.2.3 Summarized Parts List (SumBOM2)

ABC Lamp Company

Summarized Parts List, Period 2

Part	Low_Code	Gros_Req	On_Hand	Net_Req	Lead Time	Desc	Unit
1100	2	0	0	0	2	Finished shaft	Each
1200	2	0	0	0	3	6-Diameter steel plate	Each
1300	2	0	0	0	2	Hub	Each
1400	3	60	0	60	1	1/4-20 Screw	Each
1500	2	30	0	30	2	Steel holder	Each
1600	2	30	0	30	2	One-way socket	Each
1700	2	30	0	30	1	Wiring assembly	Each
2100	3	0	0	0	3	3/8 Steel tubing	Inches
2200	3	360	0	360	2	16-Gauge lamp cord	Feet
2300	3	30	0	30	1	Standard plug terminal	Each
A100	1	30	0	30	1	Socket assembly	Each
B100	1	30	50	0	1	Base assembly	Each
LA01	0	50	20	30	2	Lamp LA	Each
S100	1	30	0	30	2	Black shade	Each

The data set SumBOM2 displayed in Output 3.2.3 contains the sorted summarized parts list for the production plan, in which 50 units of 'Lamp LA' are planned (say, in period 2). Based on the input data, 'Lamp LA' has 20 units and 'Base assembly' has 50 units currently on hand. The gross requirements of 30 units for 'B100', 'S100', and 'A100' are from the net requirement of 'LA01' (computed as Gros_Req − On_Hand). Since item 'B100' has 50 units on hand, which is more than its gross requirement, the net requirement of this item is 0. This implies that the gross requirements for '1100', '1200', and '1300' (components of the item 'B100') are all 0. However, the gross requirement for item '1400', which is also a component of 'B100', is not 0 but 60. This is due to the fact that item '1400' is also used in item '1500'.

The following code uses the Indented BOM data set produced in the previous invocation of PROC BOM to create a data set that can be used by the NETDRAW procedure.

```
  /* Prepare the data set for running NETDRAW */
data IndBOM2a(drop=Part_ID);
   set IndBOM2;
   Paren_ID=Part_ID;
run;

data IndBOM2b;
   set IndBOM2(keep=Paren_ID Part_ID);
run;

data TreBOM2;
   set IndBOM2a IndBOM2b;
   LTnQP = put(LeadTime, f3.)||"  "||put(QtyPer, f3.);
run;
```

PROC NETDRAW is invoked with the NODISPLAY option to create a data set (Layout2) that contains all the layout information for the tree structure. The OUT= option specifies the name of the layout data set:

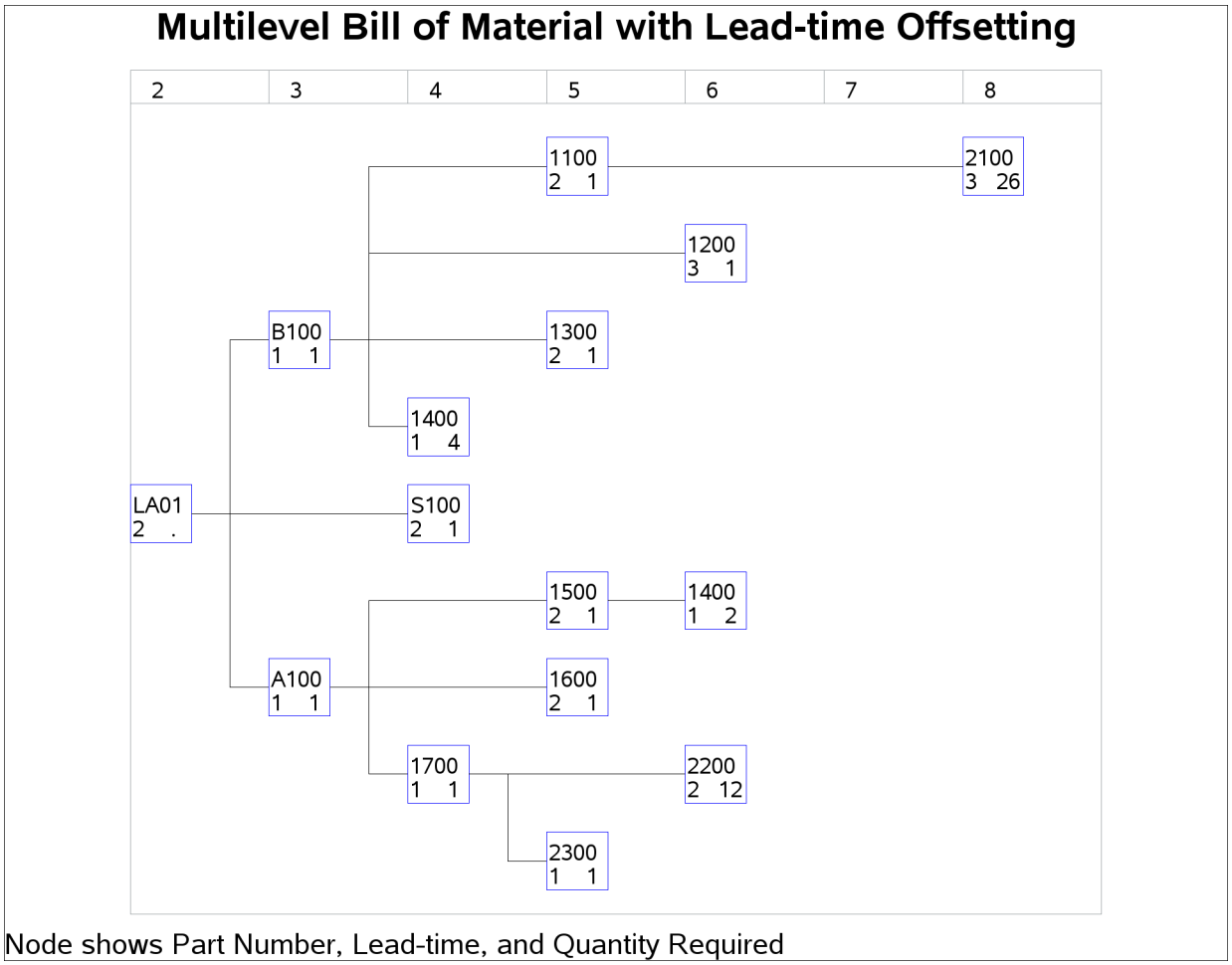
```
  /* Specify graphics options */
title h=4pct j=c 'Multilevel Bill of Material with Lead-time Offsetting';
footnote h=3pct j=l
         'Node shows Part Number, Lead-time, and Quantity Required';
pattern1 v=e c=blue;

  /* Get the layout information for the BOM tree */
proc netdraw data=TreBOM2( where=(Paren_ID NE .) )
             out=Layout2 nodisplay;
   actnet / act=Paren_ID succ=Part_ID id=(_Part_ LTnQP Tot_Lead)
            ybetween=3 xbetween=15 tree
            rectilinear nodefid nolabel;
run;
```

In the next invocation, PROC NETDRAW uses a modified layout of the nodes to produce a diagram where the nodes are aligned according to the total lead time. The resulting tree diagram is shown in Output 3.2.4. Refer to Chapter 9, "The NETDRAW Procedure" (*SAS/OR User's Guide: Project Management*), for details about the NETDRAW procedure.

```
  /* Lead time offset the X coordinate of each node */
data TreBOM2a;
   set Layout2;
   if _seq_ = 0;
```

```
   drop _seq_;
   _x_ = Tot_Lead;
run;

  /* Display the BOM tree with lead-time offsetting */
proc netdraw data=TreBOM2a out=NetOUT;;
   actnet / id=(_Part_ LTnQP)
            ctext=black htext=3 carcs=black
            align=Tot_Lead frame pcompress
            xbetween=15 ybetween=3
            arrowhead=0 rectilinear nodefid nolabel;
run;
```

Output 3.2.4 Bill of Material Diagram with Lead-time Offsetting



Example 3.3: Bill of Material with Scrap Factor Information

As in the introductory example described in the section "Getting Started: BOM Procedure" on page 17, this example uses two data files, PMaster3 and ParComp3, as input data sets for the BOM procedure. The PMaster3 data set, shown in Output 3.3.1, lists the part master records for all items of the ABC Lamp Company. The Part and Desc variables contain the part number and description, respectively. The Unit

and LeadTime variables contain the unit of measure and lead time information for the item identified by the Part variable. The ParComp3 data set (shown in Output 3.3.2) lists all product structure records in the company. The Parent and the Component variables contain the part numbers for the parent item and the component, respectively. The QtyPer, Fscrap, and LTOff variables contain the quantity per assembly, scrap factor, and lead-time offset information, respectively, for the relationship identified by the Parent and Component variables. The SDate and EDate variables are the start and end dates for the *bill of material effectivity dates*. The effectivity dates are used to determine when a component is active as a part of the bill of material. In this example, item '1700' uses component '2200' until the end of the 7th of April, 2001. Starting on April 8, 2001, item '1700' uses component '2210' instead. Refer to Landvater and Gray (1989) for more information about bill of material effectivity dates.

Output 3.3.1 Part Master Data Set (PMaster3)

ABC Lamp Company

Part Master Records

Part	Desc	Unit	Lead Time
1100	Finished shaft	Each	2
1200	6-Diameter steel plate	Each	3
1300	Hub	Each	2
1400	1/4-20 Screw	Each	1
1500	Steel holder	Each	2
1600	One-way socket	Each	2
1700	Wiring assembly	Each	1
2100	3/8 Steel tubing	Inches	3
2200	16-Gauge lamp cord	Feet	2
2210	14-Gauge lamp cord	Feet	2
2300	Standard plug terminal	Each	1
A100	Socket assembly	Each	1
B100	Base assembly	Each	1
LA01	Lamp LA	Each	2
S100	Black shade	Each	2

Output 3.3.2 Product Structure Data Set (ParComp3)

ABC Lamp Company

Product structure Records

Parent	Component	Qty Per	Fscrap	LTOff	SDate	EDate
LA01	B100	1	.	.	.	.
	S100	1	.	.	.	.
	A100	1	.	2	.	.
B100	1100	1	.	.	.	.
	1200	1	.	.	.	.
	1300	1	.	1	.	.
	1400	4	.	3	.	.
A100	1500	1	.	.	.	.
	1600	1	.	.	.	.
	1700	1	.	.	.	.
1100	2100	26	0.2	.	.	.
1500	1400	2	.	.	.	.
1700	2200	12	0.1	.	.	07APR2001
	2210	12	0.1	.	08APR2001	.
	2300	1	.	.	.	.

The following code invokes PROC BOM to produce the indented bill of material and the summarized parts list.

```
  /* Create the indented BOM with lead time */
proc bom data=ParComp3 pmdata=PMaster3
         out=IndBOM3 summaryout=SumBOM3;
   structure / part=Part
               leadtime=LeadTime
               parent=Parent
               component=Component
               quantity=QtyPer
               factor=Fscrap
               offset=LTOff
               id=(Desc Unit)
               rid=(SDate EDate);
run;
```

The indented bill of material in Output 3.3.3 is similar to the one displayed in Output 3.2.2, with additional fields for scrap factor, lead-time offset, total offset, and the start and end effectivity dates. The values of scrap factor, lead-time offset and the effectivity dates are carried from the product structure input data set shown in Output 3.3.2. The value of the total offset (denoted by the Tot_Off variable) is determined by the procedure as the total lead-time offset accumulated from the end item identified by the _Prod_ variable to the item identified by the _Part_ variable. In addition, the value of the quantity per product (denoted by the Qty_Prod variable) for each record of this indented bill of material has been increased by the scrap factor to account for anticipated loss within the manufacture of the product '`LA01`'. See the section "Indented BOM Data Set" on page 32 for information about determining the quantity per product when scrap factor is in effect.

Output 3.3.3 Indented Bill of Material with Scrap Factor (IndBOM3)

ABC Lamp Company

Indented Bill of Material, Part LA01

Obs	_Level_	_Parent_	_Part_	Desc	Qty Per	Fscrap	Qty_Prod
1	0		LA01	Lamp LA	.	.	1
2	1	LA01	B100	Base assembly	1	0.0	1
3	2	B100	1100	Finished shaft	1	0.0	1
4	3	1100	2100	3/8 Steel tubing	26	0.2	26
5	2	B100	1200	6-Diameter steel plate	1	0.0	1
6	2	B100	1300	Hub	1	0.0	1
7	2	B100	1400	1/4-20 Screw	4	0.0	4
8	1	LA01	S100	Black shade	1	0.0	1
9	1	LA01	A100	Socket assembly	1	0.0	1
10	2	A100	1500	Steel holder	1	0.0	1
11	3	1500	1400	1/4-20 Screw	2	0.0	2
12	2	A100	1600	One-way socket	1	0.0	1
13	2	A100	1700	Wiring assembly	1	0.0	1
14	3	1700	2200	16-Gauge lamp cord	12	0.1	12
15	3	1700	2210	14-Gauge lamp cord	12	0.1	12
16	3	1700	2300	Standard plug terminal	1	0.0	1

Obs	Unit	Lead Time	Tot_Lead	LTOff	Tot_Off	SDate	EDate	_Prod_
1	Each	2	2	.	0	.	.	LA01
2	Each	1	3	0	0	.	.	LA01
3	Each	2	5	0	0	.	.	LA01
4	Inches	3	8	0	0	.	.	LA01
5	Each	3	6	0	0	.	.	LA01
6	Each	2	5	1	1	.	.	LA01
7	Each	1	4	3	3	.	.	LA01
8	Each	2	4	0	0	.	.	LA01
9	Each	1	3	2	2	.	.	LA01
10	Each	2	5	0	2	.	.	LA01
11	Each	1	6	0	2	.	.	LA01
12	Each	2	5	0	2	.	.	LA01
13	Each	1	4	0	2	.	.	LA01
14	Feet	2	6	0	2	.	07APR2001	LA01
15	Feet	2	6	0	2	08APR2001	.	LA01
16	Each	1	5	0	2	.	.	LA01

The summarized parts list, which has been sorted by the _Part_ variable, is displayed in Output 3.3.4. Comparing it with the summarized parts list displayed in Output 3.1.3, you can see the impact of scrap factor on the gross requirement (and hence, the net requirement). Moreover, the summarized parts list data set shown in Output 3.3.4 contains a record for each of the items '2200' and '2210', and the gross requirements of these two items are not affected by the bill of material effectivity dates.

Output 3.3.4 Summarized Parts List (SumBOM3)

ABC Lamp Company

Summarized Parts List, Period 3

Part	Low_Code	Gros_Req	On_Hand	Net_Req	Lead Time	Desc	Unit
1100	2	1.0	0	1.0	2	Finished shaft	Each
1200	2	1.0	0	1.0	3	6-Diameter steel plate	Each
1300	2	1.0	0	1.0	2	Hub	Each
1400	3	6.0	0	6.0	1	1/4-20 Screw	Each
1500	2	1.0	0	1.0	2	Steel holder	Each
1600	2	1.0	0	1.0	2	One-way socket	Each
1700	2	1.0	0	1.0	1	Wiring assembly	Each
2100	3	31.2	0	31.2	3	3/8 Steel tubing	Inches
2200	3	13.2	0	13.2	2	16-Gauge lamp cord	Feet
2210	3	13.2	0	13.2	2	14-Gauge lamp cord	Feet
2300	3	1.0	0	1.0	1	Standard plug terminal	Each
A100	1	1.0	0	1.0	1	Socket assembly	Each
B100	1	1.0	0	1.0	1	Base assembly	Each
LA01	0	1.0	0	1.0	2	Lamp LA	Each
S100	1	1.0	0	1.0	2	Black shade	Each

Example 3.4: Planning Bill of Material

The previous examples dealt with the bill of material from the perspective of the manufacturing process. In this example, let us turn to another type of BOM that is often useful in planning and handling engineering charges. It is referred to as a *planning BOM*, *pseudo BOM*, *super BOM*, or *phantom BOM*.

Assume that the ABC Lamp Company sells lamps with eight different shades (either 14 inches or 15 inches, in the colors black, white, cream, or yellow), three alternate base plates (6 inches, 7 inches, or 8 inches), and two types of sockets ('One-way' and 'Three-way'). Working with all possible combinations, the company now has 48 (= 8×3×2) different final products. In other words, the ABC Lamp Company has 48 different end items to forecast and plan. This might not sound too complicated; however, this example is a simplified case. Manufacturing companies producing automobiles, computers, or machine tools can easily make over 5,000 different final products. So many final products are obviously impossible to forecast accurately and plan with any hope of validity. However, a reasonable forecast for total lamp sales could be made, for example, 10,000 per week. Some items (such as '`1100 Finished shaft`') are common to all configurations. Although these items are produced independently of one another, they can be grouped as common items on the BOM for administrative purposes. The new item that is created to group all common items is never stocked and hence it is called a *phantom item*. Phantom items are generally assigned 0 lead times and lot-for-lot order quantity. Besides the phantom item for grouping common items, other phantom items are created for planning different options. These items are referred to as *model items*. This process is known as the *modularized bill of material construction*.

The following SAS code creates three data sets: The Phantom4 data set contains the generic item '`LAXX`' and all phantom items. The data set Option4 lists all options of each model item that are not already in the Part Master data set PMaster3 (shown in Output 3.3.1). Finally, the ParComp4 data set contains the re-grouped product structure information.

```
   /* Generic and phantom part master data */

data Phantom4;
   input Part      $8.
         Desc      $24.
         Req       8.0
         Unit      $8.
         LeadTime  4.0
         ;
datalines;
LAXX    Lamp LA                    10000 Each      3
4000    Common parts                   . Each      0
A10X    Socket assembly options        . Each      0
B10X    Base assembly options          . Each      0
S10X    Shade options                  . Each      0
;


  /* Additional optional and alternative parts */

data Option4;
   input Part      $8.
         Desc      $24.
         Req       8.0
         Unit      $8.
         LeadTime  4.0
         ;
datalines;
A101    Three-way socket assem.        . Each      1
B101    7in Base assembly              . Each      1
B102    8in Base assembly              . Each      1
S101    14in White shade               . Each      2
S102    14in Cream shade               . Each      2
S103    14in Yellow shade              . Each      2
S104    15in Black shade               . Each      2
S105    15in White shade               . Each      2
S106    15in Cream shade               . Each      2
S107    15in Yellow shade              . Each      2
1201    7-Diameter steel plate         . Each      3
1202    8-Diameter steel plate         . Each      3
1601    Three-way socket               . Each      2
;

  /* Parent-component relationship data */

data ParComp4;
   input Parent    $8.
         Component $8.
         QtyPer    8.2
         ;
datalines;
LAXX    4000        1.00
LAXX    B10X        1.00
```

```
LAXX    S10X        1.00
LAXX    A10X        1.00
4000    1100        1.00
4000    1300        1.00
4000    1400        4.00
4000    1500        1.00
4000    1700        1.00
B10X    B100        0.32
B10X    B101        0.41
B10X    B102        0.33
S10X    S100        0.07
S10X    S101        0.18
S10X    S102        0.24
S10X    S103        0.10
S10X    S104        0.06
S10X    S105        0.14
S10X    S106        0.22
S10X    S107        0.10
A10X    A100        0.11
A10X    A101        0.92
B100    1200        1.00
B101    1201        1.00
B102    1202        1.00
A100    1600        1.00
A101    1601        1.00
1100    2100       26.00
1500    1400        2.00
1700    2200       12.00
1700    2300        1.00
;
```

The item identified as '4000' is the phantom item for common items '1100', '1300', '1400', '1500', and '1700'. Each available option is structured into a model item with a quantity per assembly (identified as the QtyPer variable) that represents its forecast popularity or option percentage. Note that the total percentage of each option in this example is more than 100 percent (for example, the total percentage of the 'A10X: Socket assembly options' is $0.11+0.92 = 1.03$). This extra percentage is used to cover the uncertainty of the exact percentage split. Using this procedure to cover possible high side demand for each option is called *option overplanning* (Fogarty, Blackstone, and Hoffmann 1991).

The new Part Master data set, PMaster4, combines the generic and phantom item data, the additional options data, and the old Part Master file shown in Output 3.3.1. The SAS code to accomplish this is as follows:

```
  /* Append the old part master data to the new */
  /* phantom and optional part data set         */

data PMaster4;
   set Phantom4
       Option4
       PMaster3(where=(Part NE 'LA01' AND Part NE '2210'));
run;

proc sort data=PMaster4;
   by Part;
run;
```

The following code invokes PROC BOM to generate the planning bill of material and the summarized parts list from the modularized product structure data and the expanded part master file.

```
   /* Generate the Indented BOM and Summarized Parts data sets */
proc bom data=ParComp4 pmdata=PMaster4
         out=IndBOM4 summaryout=SumBOM4;
   structure / part=Part
               requirement=Req
               leadtime=LeadTime
               parent=Parent
               component=Component
               quantity=QtyPer
               id=(Desc Unit);
run;
```

The indented bill of material is shown in Output 3.4.1. Output 3.4.2 lists the quantity of each item that is needed to build 10,000 lamps in period 4, sorted by the _Part_ variable.

Output 3.4.1 Planning Bill of Material with Option Overplanning (IndBOM4)

ABC Lamp Company

Indented Bill of Material, Part LAXX

Level	_Parent_	_Part_	Desc	Qty Per	Qty_Prod	Unit	Lead Time	Total Lead_	_Prod_
0		LAXX	Lamp LA	.	1.00	Each	3	3	LAXX
1	LAXX	4000	Common parts	1.00	1.00	Each	0	3	LAXX
2	4000	1100	Finished shaft	1.00	1.00	Each	2	5	LAXX
3	1100	2100	3/8 Steel tubing	26.00	26.00	Inches	3	8	LAXX
2	4000	1300	Hub	1.00	1.00	Each	2	5	LAXX
2	4000	1400	1/4-20 Screw	4.00	4.00	Each	1	4	LAXX
2	4000	1500	Steel holder	1.00	1.00	Each	2	5	LAXX
3	1500	1400	1/4-20 Screw	2.00	2.00	Each	1	6	LAXX
2	4000	1700	Wiring assembly	1.00	1.00	Each	1	4	LAXX
3	1700	2200	16-Gauge lamp cord	12.00	12.00	Feet	2	6	LAXX
3	1700	2300	Standard plug terminal	1.00	1.00	Each	1	5	LAXX
1	LAXX	B10X	Base assembly options	1.00	1.00	Each	0	3	LAXX
2	B10X	B100	Base assembly	0.32	0.32	Each	1	4	LAXX
3	B100	1200	6-Diameter steel plate	1.00	0.32	Each	3	7	LAXX
2	B10X	B101	7in Base assembly	0.41	0.41	Each	1	4	LAXX
3	B101	1201	7-Diameter steel plate	1.00	0.41	Each	3	7	LAXX
2	B10X	B102	8in Base assembly	0.33	0.33	Each	1	4	LAXX
3	B102	1202	8-Diameter steel plate	1.00	0.33	Each	3	7	LAXX
1	LAXX	S10X	Shade options	1.00	1.00	Each	0	3	LAXX
2	S10X	S100	Black shade	0.07	0.07	Each	2	5	LAXX
2	S10X	S101	14in White shade	0.18	0.18	Each	2	5	LAXX
2	S10X	S102	14in Cream shade	0.24	0.24	Each	2	5	LAXX
2	S10X	S103	14in Yellow shade	0.10	0.10	Each	2	5	LAXX
2	S10X	S104	15in Black shade	0.06	0.06	Each	2	5	LAXX
2	S10X	S105	15in White shade	0.14	0.14	Each	2	5	LAXX
2	S10X	S106	15in Cream shade	0.22	0.22	Each	2	5	LAXX
2	S10X	S107	15in Yellow shade	0.10	0.10	Each	2	5	LAXX
1	LAXX	A10X	Socket assembly options	1.00	1.00	Each	0	3	LAXX
2	A10X	A100	Socket assembly	0.11	0.11	Each	1	4	LAXX
3	A100	1600	One-way socket	1.00	0.11	Each	2	6	LAXX
2	A10X	A101	Three-way socket assem.	0.92	0.92	Each	1	4	LAXX
3	A101	1601	Three-way socket	1.00	0.92	Each	2	6	LAXX

Output 3.4.2 Summarized Parts List (SumBOM4)

ABC Lamp Company

Summarized Parts List, Period 4

Part	Low_Code	Req	On_Hand	Net_Req	Lead Time	Desc	Unit
1100	2	10000	0	10000	2	Finished shaft	Each
1200	3	3200	0	3200	3	6-Diameter steel plate	Each
1201	3	4100	0	4100	3	7-Diameter steel plate	Each
1202	3	3300	0	3300	3	8-Diameter steel plate	Each
1300	2	10000	0	10000	2	Hub	Each
1400	3	60000	0	60000	1	1/4-20 Screw	Each
1500	2	10000	0	10000	2	Steel holder	Each
1600	3	1100	0	1100	2	One-way socket	Each
1601	3	9200	0	9200	2	Three-way socket	Each
1700	2	10000	0	10000	1	Wiring assembly	Each
2100	3	260000	0	260000	3	3/8 Steel tubing	Inches
2200	3	120000	0	120000	2	16-Gauge lamp cord	Feet
2300	3	10000	0	10000	1	Standard plug terminal	Each
4000	1	10000	0	10000	0	Common parts	Each
A100	2	1100	0	1100	1	Socket assembly	Each
A101	2	9200	0	9200	1	Three-way socket assem.	Each
A10X	1	10000	0	10000	0	Socket assembly options	Each
B100	2	3200	0	3200	1	Base assembly	Each
B101	2	4100	0	4100	1	7in Base assembly	Each
B102	2	3300	0	3300	1	8in Base assembly	Each
B10X	1	10000	0	10000	0	Base assembly options	Each
LAXX	0	10000	0	10000	3	Lamp LA	Each
S100	2	700	0	700	2	Black shade	Each
S101	2	1800	0	1800	2	14in White shade	Each
S102	2	2400	0	2400	2	14in Cream shade	Each
S103	2	1000	0	1000	2	14in Yellow shade	Each
S104	2	600	0	600	2	15in Black shade	Each
S105	2	1400	0	1400	2	15in White shade	Each
S106	2	2200	0	2200	2	15in Cream shade	Each
S107	2	1000	0	1000	2	15in Yellow shade	Each
S10X	1	10000	0	10000	0	Shade options	Each

Example 3.5: Modular Bill of Material

The previous example illustrated how the bill of material can be arranged in product modules or options in order to obtain better forecasting, master production scheduling, and planning. There are also a number of other reasons for using *modular bill of material*. One reason is that the task of maintaining modularized bill of material is much simpler. The only bill of material that have to be maintained are the modules; there is no need to maintain enormous numbers of unique product configurations (Landvater and Gray 1989). You can easily construct a bill of material for a particular product configuration from the modular bill of material, if necessary. This example shows how to build modular bill of material for the 48 product configurations as described in Example 3.4. It also demonstrates how to construct a final-product-oriented bill of material from the modularized bill of material.

The ParComp5 data set contains the new product structure information. In this data set, the phantom items 'A10X', 'B10X', and 'S10X' are used as place holders for product modules: 'A100', 'A101', 'B100', ..., 'S107'. The following code first creates the new Product Structure data set, ParComp5. It then invokes PROC BOM with the Part Master data set, PMaster4 (described in Example 3.4), and the new Product Structure data set, ParComp5, to construct modular bill of material for the production line of the company. It also displays the modularized indented bill of material.

```
/* Product Structure data */
data ParComp5;
   format Parent $8. Component $8. QtyPer 8.2 ;
   input Parent $ Component $ QtyPer;
datalines;
LAXX    4000        1.00
LAXX    B10X        1.00
LAXX    S10X        1.00
LAXX    A10X        1.00
4000    1100        1.00
4000    1300        1.00
4000    1400        4.00
4000    1500        1.00
4000    1700        1.00
B100    1200        1.00
B101    1201        1.00
B102    1202        1.00
A100    1600        1.00
A101    1601        1.00
1100    2100       26.00
1500    1400        2.00
1700    2200       12.00
1700    2300        1.00
;

/* Generate the indented BOM data set */
proc bom data=ParComp5 pmdata=PMaster4 out=IndBOM5;
   structure / part=Part
               leadtime=LeadTime
               parent=Parent
               component=Component
               quantity=QtyPer
               id=(Desc Unit);
run;
```

The Indented BOM data set, IndBOM5, is shown in Output 3.5.1. This output data set contains a pseudo bill of material for the pseudo end item 'LAXX' and 13 modules (modular bill of material) for items 'A100', 'A101', 'B100', 'B101', 'B102', and 'S100' ..., 'S107', respectively.

Output 3.5.1 Modularized Indented Bill of Material (IndBOM5)

ABC Lamp Company

Indented Bill of Material

Level	_Parent_	_Part_	Desc	Qty Per	Qty_Prod	Unit	Lead Time	Total_Lead	_Prod_
0		A100	Socket assembly	.	1.00	Each	1	1	A100
1	A100	1600	One-way socket	1.00	1.00	Each	2	3	A100
0		A101	Three-way socket assem.	.	1.00	Each	1	1	A101
1	A101	1601	Three-way socket	1.00	1.00	Each	2	3	A101
0		B100	Base assembly	.	1.00	Each	1	1	B100
1	B100	1200	6-Diameter steel plate	1.00	1.00	Each	3	4	B100
0		B101	7in Base assembly	.	1.00	Each	1	1	B101
1	B101	1201	7-Diameter steel plate	1.00	1.00	Each	3	4	B101
0		B102	8in Base assembly	.	1.00	Each	1	1	B102
1	B102	1202	8-Diameter steel plate	1.00	1.00	Each	3	4	B102
0		LAXX	Lamp LA	.	1.00	Each	3	3	LAXX
1	LAXX	4000	Common parts	1.00	1.00	Each	0	3	LAXX
2	4000	1100	Finished shaft	1.00	1.00	Each	2	5	LAXX
3	1100	2100	3/8 Steel tubing	26.00	26.00	Inches	3	8	LAXX
2	4000	1300	Hub	1.00	1.00	Each	2	5	LAXX
2	4000	1400	1/4-20 Screw	4.00	4.00	Each	1	4	LAXX
2	4000	1500	Steel holder	1.00	1.00	Each	2	5	LAXX
3	1500	1400	1/4-20 Screw	2.00	2.00	Each	1	6	LAXX
2	4000	1700	Wiring assembly	1.00	1.00	Each	1	4	LAXX
3	1700	2200	16-Gauge lamp cord	12.00	12.00	Feet	2	6	LAXX
3	1700	2300	Standard plug terminal	1.00	1.00	Each	1	5	LAXX
1	LAXX	B10X	Base assembly options	1.00	1.00	Each	0	3	LAXX
1	LAXX	S10X	Shade options	1.00	1.00	Each	0	3	LAXX
1	LAXX	A10X	Socket assembly options	1.00	1.00	Each	0	3	LAXX
0		S100	Black shade	.	1.00	Each	2	2	S100
0		S101	14in White shade	.	1.00	Each	2	2	S101
0		S102	14in Cream shade	.	1.00	Each	2	2	S102
0		S103	14in Yellow shade	.	1.00	Each	2	2	S103
0		S104	15in Black shade	.	1.00	Each	2	2	S104
0		S105	15in White shade	.	1.00	Each	2	2	S105
0		S106	15in Cream shade	.	1.00	Each	2	2	S106
0		S107	15in Yellow shade	.	1.00	Each	2	2	S107

The following SAS code constructs an indented bill of material for the product '`LA01`', which is configured with '`A100`' (`One-way socket assembly`), '`B100`' (`6in Base assembly`), and '`S100`' (`14in Black shade`). It first uses the %BOMTSAE SAS macro to create a bill of material that is the same as the bill for the item '`LAXX`', except the pseudo item '`B10X`' is replaced by the modular bill for the item '`B100`'. It then uses the %BOMTREP SAS macro to replace the pseudo items '`S10X`' and '`A10X`' with the bill of material for '`S100`' and '`A100`'. Note that the %BOMTSAE macro refers to the part master file PMaster3 (shown in Output 3.3.1) for the master data of the item '`LA01`'. See the section "Transactional Macros" on page 104 for details about these SAS macros.

```
/* Copy LAXX to LA01 with B10X replaced by B100 */
%bomtsae(root='LA01', sameas='LAXX', except='B10X', repby='B100',
      in=IndBOM5, pmdata=PMaster3, part=Part, quantity=QtyPer,
      leadtime=LeadTime, id=Desc Unit, out=BomOut5);

/* Merge the bill of material for S100 to the new BOM */
data BomOut51;
set BomOut5
    IndBOM5(where=(_Prod_='S100'));
run;

/* Replace S10X with S100 */
%bomtrep(root='S10X', repby='S100', in=BomOut51, quantity=QtyPer,
      leadtime=LeadTime, id=Desc Unit, del=1, out=BomOut52);

/* Merge the bill of material for A100 to the new BOM */
data BomOut53;
set BomOut52
    IndBOM5(where=(_Prod_='A100'));
run;

/* Replace A10X with A100 */
%bomtrep(root='A10X', repby='A100', in=BomOut53, quantity=QtyPer,
      leadtime=LeadTime, id=Desc Unit, del=1, out=BomOut5);
```

The indented bill of material for item '`LA01`' is displayed in Output 3.5.2. It is the same as the bill of material displayed in Output 3.2.2 (which is created directly by PROC BOM), except for the common items such as '`1100`', '`1300`', '`1400`', '`1500`', and '`1700`', which are regrouped under the phantom item '`4000`'.

Output 3.5.2 Indented Bill of Material for Product 'LA01' (BomOut5)

ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Desc	Qty_Per	Qty_Per	Unit	LTime	LTOff_Lead	_Prod_
0		LA01	Lamp LA	.	1.00	Each	2	2	LA01
1	LA01	4000	Common parts	1.00	1.00	Each	0	2	LA01
2	4000	1100	Finished shaft	1.00	1.00	Each	2	4	LA01
3	1100	2100	3/8 Steel tubing	26.00	26.00	Inches	3	7	LA01
2	4000	1300	Hub	1.00	1.00	Each	2	4	LA01
2	4000	1400	1/4-20 Screw	4.00	4.00	Each	1	3	LA01
2	4000	1500	Steel holder	1.00	1.00	Each	2	4	LA01
3	1500	1400	1/4-20 Screw	2.00	2.00	Each	1	5	LA01
2	4000	1700	Wiring assembly	1.00	1.00	Each	1	3	LA01
3	1700	2200	16-Gauge lamp cord	12.00	12.00	Feet	2	5	LA01
3	1700	2300	Standard plug terminal	1.00	1.00	Each	1	4	LA01
1	LA01	A100	Socket assembly	1.00	1.00	Each	1	3	LA01
2	A100	1600	One-way socket	1.00	1.00	Each	2	5	LA01
1	LA01	B100	Base assembly	1.00	1.00	Each	1	3	LA01
2	B100	1200	6-Diameter steel plate	1.00	1.00	Each	3	6	LA01
1	LA01	S100	Black shade	1.00	1.00	Each	2	4	LA01

Example 3.6: Bill of Material with Repeated Relationships

As briefly described in the section "Product Structure Data Set" on page 30, sometimes it is necessary to structure a given component into a parent item more than once because it is used at different places, at different times in the process, or in different operations. The Product Structure data set shown in Output 3.6.1 is an example of this situation. Because of the different points of use (denoted by the PointUse variable) and lead-time offsets (denoted by the LTOff variable) involved in its two relationships, item '1400' (1/4-20 Screw) cannot be structured just once into the parent item 'B100' (Base assembly). The Line variable in this data set denotes the *line sequence number*, which is part of the unique *key* in the product structure record. Although it is not necessary in the BOM procedure, some software packages will not allow two relationships with the same parent and component. However, when unique line sequence numbers are assigned, a given component can be structured into a parent item any number of times. Line sequence numbers are sometimes used to sequence components in the desired order on a bill of material, frequently in order of use. They are also used on occasion to refer to the *balloon* or *find* number on engineer drawings (Clement, Coldrick, and Sari 1992).

Output 3.6.1 Product Structure Data Set with Repeated Relationships

ABC Lamp Company

Product Structure Records

Parent	Component	Qty Per	Fscrap	Line	Point Use	LTOff
LA01	B100	1	.	010	SA2	0
	S100	1	.	015	SA2	0
	A100	1	.	020	SA5	15
B100	1100	1	.	010	SA4A	0
	1200	1	.	020	SA4A	0
	1400	4	0.25	110	SA4A	0
	1300	1	.	120	SA4B	20
	1400	4	0.50	215	SA4B	20
A100	1500	1	.	100	SA3	0
	1600	1	.	110	SA3	0
	1700	1	.	120	SA5	0
1100	2100	26	0.20		SA9B	0
1500	1400	2	.		SA7	0
1700	2200	12	0.10	010	SA5	0
	2300	1	.	030	SA5	5

The following code invokes PROC BOM with this Product Structure data set, ParComp6, and the Part Master data set, PMaster3 (shown in Output 3.3.1), to create the indented bill of material for the product '`LA01`'. The Line and PointUse variables specified in the RID= option can be used to distinguish the two relationships with the same parent item '`B100`' and components '`1400`' (observations 6 and 8) in the indented BOM data set as displayed in Output 3.6.2.

```
  /* Create the indented BOM */
proc bom data=ParComp6 pmdata=PMaster3 out=IndBOM6a;
   structure / part=Part
               leadtime=LeadTime
               parent=Parent
               component=Component
               quantity=QtyPer
               factor=Fscrap
               offset=LTOff
               id=(Desc Unit)
               rid=(Line PointUse)
               enditem=("LA01");
run;
```

Output 3.6.2 Indented Bill of Material with Repeated Relationships

ABC Lamp Company

Indented Bill of Material, Part LA01

Obs	_Level_	_Parent_	_Part_	Desc	_Qty_	_Fscrap_	_QtyPer_	Unit	_LeadTime_	_LTOffset_	_TotLTOffset_	_Level_	_Position_	_Prod_
1	0		LA01	Lamp LA	.	.	1	Each	2	2	.	0		LA01
2	1	LA01	B100	Base assembly	1	0.00	1	Each	1	3	0	0	010	SA2 LA01
3	2	B100	1100	Finished shaft	1	0.00	1	Each	2	5	0	0	010	SA4A LA01
4	3	1100	2100	3/8 Steel tubing	26	0.20	26	Inches	3	8	0	0		SA9B LA01
5	2	B100	1200	6-Diameter steel plate	1	0.00	1	Each	3	6	0	0	020	SA4A LA01
6	2	B100	1400	1/4-20 Screw	4	0.25	4	Each	1	4	0	0	110	SA4A LA01
7	2	B100	1300	Hub	1	0.00	1	Each	2	5	20	20	120	SA4B LA01
8	2	B100	1400	1/4-20 Screw	4	0.50	4	Each	1	4	20	20	215	SA4B LA01
9	1	LA01	S100	Black shade	1	0.00	1	Each	2	4	0	0	015	SA2 LA01
10	1	LA01	A100	Socket assembly	1	0.00	1	Each	1	3	15	15	020	SA5 LA01
11	2	A100	1500	Steel holder	1	0.00	1	Each	2	5	0	15	100	SA3 LA01
12	3	1500	1400	1/4-20 Screw	2	0.00	2	Each	1	6	0	15		SA7 LA01
13	2	A100	1600	One-way socket	1	0.00	1	Each	2	5	0	15	110	SA3 LA01
14	2	A100	1700	Wiring assembly	1	0.00	1	Each	1	4	0	15	120	SA5 LA01
15	3	1700	2200	16-Gauge lamp cord	12	0.10	12	Feet	2	6	0	15	010	SA5 LA01
16	3	1700	2300	Standard plug terminal	1	0.00	1	Each	1	5	5	20	030	SA5 LA01

The BOM procedure can create bill of material with two or more identical relationships in a given parent item, if necessary. The specification of the keyword `KEEP` for the DUPLICATE= option tells PROC BOM to sequence components in the same order as they are listed in the Product Structure data set while constructing the indented bill of material. The following SAS code invokes PROC BOM to create a bill of material with identical relationships, displayed in Output 3.6.3. Note that the RID= option is not specified in this invocation of PROC BOM: the two relationships with the same parent item '`B100`' and component '`1400`' are identical.

```
/* Create the indented BOM */
proc bom data=ParComp6 pmdata=PMaster3 out=IndBOM6b
      duplicate=KEEP;
structure / part=Part
            leadtime=LeadTime
            parent=Parent
            component=Component
            quantity=QtyPer
            factor=Fscrap
            id=(Desc Unit)
            enditem=("LA01");
run;
```

Output 3.6.3 Indented Bill of Material with Identical Relationships

ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Desc	Qty Per	Fscrap	Qty_Prod	Unit	LT Level	LT Total	_Prod_
0		LA01	Lamp LA	.	.	1	Each	2	2	LA01
1	LA01	B100	Base assembly	1	0.00	1	Each	1	3	LA01
2	B100	1100	Finished shaft	1	0.00	1	Each	2	5	LA01
3	1100	2100	3/8 Steel tubing	26	0.20	26	Inches	3	8	LA01
2	B100	1200	6-Diameter steel plate	1	0.00	1	Each	3	6	LA01
2	B100	1400	1/4-20 Screw	4	0.25	4	Each	1	4	LA01
2	B100	1300	Hub	1	0.00	1	Each	2	5	LA01
2	B100	1400	1/4-20 Screw	4	0.50	4	Each	1	4	LA01
1	LA01	S100	Black shade	1	0.00	1	Each	2	4	LA01
1	LA01	A100	Socket assembly	1	0.00	1	Each	1	3	LA01
2	A100	1500	Steel holder	1	0.00	1	Each	2	5	LA01
3	1500	1400	1/4-20 Screw	2	0.00	2	Each	1	6	LA01
2	A100	1600	One-way socket	1	0.00	1	Each	2	5	LA01
2	A100	1700	Wiring assembly	1	0.00	1	Each	1	4	LA01
3	1700	2200	16-Gauge lamp cord	12	0.10	12	Feet	2	6	LA01
3	1700	2300	Standard plug terminal	1	0.00	1	Each	1	5	LA01

On other occasions, you may want the bill of material to show the total quantity of each component used by a given parent item without any reference to the order, time, or place of use. To do this, you can specify the keyword *COMBINE* for the DUPLICATE= option. This is the default behavior of the DUPLICATE= option. The SAS code to accomplish this is as follows:

```
/* Create the indented BOM */
proc bom data=ParComp6 pmdata=PMaster3 out=IndBOM6c
      duplicate=COMBINE;
structure / part=Part
            leadtime=LeadTime
            parent=Parent
            component=Component
            quantity=QtyPer
            factor=Fscrap
            id=(Desc Unit)
            enditem=("LA01");
run;
```

The indented bill of material is displayed in Output 3.6.4. Note that the two identical relationships of the parent '`B100`' and the component '`1400`' are collapsed into one relationship. See the section "Product Structure Data Set" on page 30 for details about combining identical parent-component relationships.

Output 3.6.4 Indented Bill of Material with Identical Relationships Collapsed

ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Desc	Qty Per	Fsc	Qty_Prod	Unit	Lead Time	Total_Lead	_Prod_
0		LA01	Lamp LA	.	.	1	Each	2	2	LA01
1	LA01	B100	Base assembly	1	0.000	1	Each	1	3	LA01
2	B100	1100	Finished shaft	1	0.000	1	Each	2	5	LA01
3	1100	2100	3/8 Steel tubing	26	0.200	26	Inches	3	8	LA01
2	B100	1200	6-Diameter steel plate	1	0.000	1	Each	3	6	LA01
2	B100	1400	1/4-20 Screw	8	0.375	8	Each	1	4	LA01
2	B100	1300	Hub	1	0.000	1	Each	2	5	LA01
1	LA01	S100	Black shade	1	0.000	1	Each	2	4	LA01
1	LA01	A100	Socket assembly	1	0.000	1	Each	1	3	LA01
2	A100	1500	Steel holder	1	0.000	1	Each	2	5	LA01
3	1500	1400	1/4-20 Screw	2	0.000	2	Each	1	6	LA01
2	A100	1600	One-way socket	1	0.000	1	Each	2	5	LA01
2	A100	1700	Wiring assembly	1	0.000	1	Each	1	4	LA01
3	1700	2200	16-Gauge lamp cord	12	0.100	12	Feet	2	6	LA01
3	1700	2300	Standard plug terminal	1	0.000	1	Each	1	5	LA01

Example 3.7: Bill of Material Verification

The previous examples illustrated how companies may have more than one type of bill of material. It is important that while the different types of BOM may differ in the way items are grouped together, they must use the same components (except the phantom items) in the same quantities, or require the same amount of materials in order to fill the same production plan. This example demonstrates some simple methods that compare the different types of BOM and detect any differences.

One way to compare two bill of material is to compare their summarized parts lists for the same production plan. The following SAS code creates a transactional data set, Trans7, which contains the new values of the gross requirements for the lamp 'LAXX' and all base assembly, shade, or socket assembly options that are described in Example 3.4. The gross requirements for the lamp 'LAXX' and options 'B100', 'S100', and 'A100' that are associated with the lamp 'LA01' are 1. All other options have 0 gross requirement. Therefore, the item 'LAXX' and all of the options listed in the data set Trans7 are master schedule items. See the section "Part Master Data Set" on page 29 for a detailed description about master schedule items.

```
/* Gross requirement transactional data set */
data Trans7;
format Part $8. Req 8.0 ;
input Part $ Req ;
datalines;
```

```
LAXX          1
B100          1
B101          0
B102          0
S100          1
S101          0
S102          0
S103          0
S104          0
S105          0
S106          0
S107          0
A100          1
A101          0
;
```

The following SAS DATA step code updates the value of the Req variable in the PMaster4 data set (described in Example 3.4) with the value of the variable in the transactional data set Trans7.

```
proc sort data=Trans7;
   by Part;
run;

/* Update the gross requirement values of the */
/* Product Structure data set                 */
data PMaster7(drop=OldReq);
merge PMaster4(rename=(Req=OldReq)) Trans7(in=in2);
by Part;

if not in2 then Req=OldReq;
run;
```

The following code invokes PROC BOM with the new Part Master data set PMaster7 and the Product Structure data set ParComp4 (described in Example 3.4). The summarized parts list is shown in Output 3.7.1.

```
   /* Generate the indented BOM and summarized parts list */
proc bom data=ParComp4 pmdata=PMaster7
         out=IndBOM7 summaryout=SumBOM7;
   structure / part=Part
               requirement=Req
               leadtime=LeadTime
               parent=Parent
               component=Component
               quantity=QtyPer
               id=(Desc Unit);
run;
```

Output 3.7.1 Summarized Parts List (SumBOM7)

ABC Lamp Company

Summarized Parts List, Period 1

Part	Low_Code	Req	On_Hand	Net_Req	Lead Time	Desc	Unit
1100	2	1	0	1	2	Finished shaft	Each
1200	3	1	0	1	3	6-Diameter steel plate	Each
1300	2	1	0	1	2	Hub	Each
1400	3	6	0	6	1	1/4-20 Screw	Each
1500	2	1	0	1	2	Steel holder	Each
1600	3	1	0	1	2	One-way socket	Each
1700	2	1	0	1	1	Wiring assembly	Each
2100	3	26	0	26	3	3/8 Steel tubing	Inches
2200	3	12	0	12	2	16-Gauge lamp cord	Feet
2300	3	1	0	1	1	Standard plug terminal	Each
4000	1	1	0	1	0	Common parts	Each
A100	2	1	0	1	1	Socket assembly	Each
A10X	1	1	0	1	0	Socket assembly options	Each
B100	2	1	0	1	1	Base assembly	Each
B10X	1	1	0	1	0	Base assembly options	Each
LAXX	0	1	0	1	3	Lamp LA	Each
S100	2	1	0	1	2	Black shade	Each
S10X	1	1	0	1	0	Shade options	Each

The summarized parts list in Output 3.7.1 and the one in Output 3.1.3 are in agreement in the values of gross and net requirements, except for those phantom items, such as '`4000 Common parts`', '`A10X Socket assembly options`', '`B10X Base assembly options`', and '`S10X Shade options`', which are not listed in Output 3.1.3.

Another way to compare two bill of material is to compare their summarized bill of material. Recall that, as discussed in Example 3.1, the summarized parts list as displayed in Output 3.1.3 is also the summarized bill of material for the item '`LA01`'. You can also use the %BOMRSUB SAS macro described in Chapter 4, "Bill of Material Postprocessing Macros," or the SAS DATA step to create the summarized bill of material from an indented bill of material. See Example 3.9 for an example of using a SAS DATA step to create a summarized bill of material from the Indented BOM data set.

The following SAS code uses the %BOMRSUB macro to create a summarized bill of material for the item '`LA01`' from the Indented BOM data set shown in Output 3.5.2. The "″qtyreq=Gros_Req specification indicates that the total requirement for the item identified by the _Part_ variable should be stored in the Gros_Req variable.

```
/* Create the summarized bill of material */
%bomrsub(root='LA01', in=BomOut5, quantity=QtyPer,
qtyreq=Gros_Req, out=BomOut7);
```

Output 3.7.2 Summarized Bill of Material (BomOut7)

ABC Lamp Company

Summarized Bill of Material, Part LA01

Part	Gros_Req	Desc	Unit
1100	1	Finished shaft	Each
1200	1	6-Diameter steel plate	Each
1300	1	Hub	Each
1400	6	1/4-20 Screw	Each
1500	1	Steel holder	Each
1600	1	One-way socket	Each
1700	1	Wiring assembly	Each
2100	26	3/8 Steel tubing	Inches
2200	12	16-Gauge lamp cord	Feet
2300	1	Standard plug terminal	Each
4000	1	Common parts	Each
A100	1	Socket assembly	Each
B100	1	Base assembly	Each
S100	1	Black shade	Each

Again, the summarized bill of material, shown in Output 3.7.2, agrees with the summarized bill of material displayed in Output 3.1.3 in the values of gross requirements, except for the phantom items '`4000 Common parts`'. Note also that the gross requirement of the finished good '`LA01`' is not listed in Output 3.7.2.

Example 3.8: Roll-Up Cost in Indented Bill of Material

As mentioned in the section "Bill of Material Explosion and Implosion" on page 35, the presence of the _Level_ variable and the order of the observations in the Indented BOM data set enables easy bill of material explosion and implosion. This example demonstrates how to calculate the costs for upper-level items using the SAS DATA step.

Suppose the ABC Lamp Company wishes to calculate the cost of the lamp '`LA01`' based on the costs of purchased items that are at the "leaf" nodes of the BOM family tree. The Indented BOM data set can be used to aggregate values assigned at the "leaf" nodes, up the BOM family tree to calculate the value at the root node, or the final product '`LA01`'.

In the following SAS code, the data set PMaster8 extends the part master file as displayed in Figure 3.1 to contain the cost for purchased items. PROC BOM is invoked with this Part Master data set and the Product Structure data set, ParComp0 as displayed in Figure 3.2 to create the Indented BOM data set, IndBOM8.

```
   /* Part master records */
data PMaster8;
   format Part $8. Desc $24. Unit $8. Cost 8.2 ;
   input Part $ Desc & Unit $ Cost;
   datalines;
1100    Finished shaft          Each         .
1200    6-Diameter steel plate  Each      9.25
1300    Hub                     Each      5.00
```

```
   1400    1/4-20 Screw            Each        0.20
   1500    Steel holder            Each         .
   1600    One-way socket          Each        3.50
   1700    Wiring assembly         Each         .
   2100    3/8 Steel tubing        Inches      0.05
   2200    16-Gauge lamp cord      Feet        0.35
   2300    Standard plug terminal  Each        0.50
   A100    Socket assembly         Each         .
   B100    Base assembly           Each         .
   LA01    Lamp LA                 Each         .
   S100    Black shade             Each        4.10
   ;

  /* Create the indented BOM */
proc bom pmdata=PMaster8 data=ParComp0 out=IndBOM8;
   structure / part=Part
               parent=Parent
               component=Component
               quantity=QtyPer
               id=(Desc Unit Cost);
run;
```

The following code first reverses the order of the observations in the Indented BOM data set, and then determines the costs for the upper-level items. Afterward, the order of the observations is restored to the original order. The indented bill of material with cost information is displayed in Output 3.8.1. In each observation, the variable Cost contains the cost for the item identified by the value of the _Part_ variable.

```
  /* Sort the indented BOM in reverse order */
proc sort data=IndBOM8(drop=_Prod_ Paren_ID);
   by descending Part_ID;
run;

  /* Roll up material cost */
data IndBOM8a;
   set IndBOM8;
   array costs[4] cost1 cost2 cost3 cost4;
   retain cost1 cost2 cost3 cost4 0;
   drop cost1 cost2 cost3 cost4;

   /* Determine the cost for the current item */
   if Cost=. then Cost=0;
   Cost = Cost + costs[_Level_+1];

   /* Roll up cost to the upper-level item */
   if _Level_ > 0 then
      costs[_Level_]=costs[_Level_]+(QtyPer*Cost);

   /* Reset roll up cost for the current level */
   costs[_Level_+1]=0;

   output;
run;
```

```
  /* Sort the indented BOM back to the original order */
proc sort data=IndBOM8a;
   by Part_ID;
run;
```

Output 3.8.1 Indented Bill of Material with Roll-Up Cost (IndBOM8a)

ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Description	Qty Per	Qty_	Unit	Costs
0		LA01	Lamp LA	.	1	Each	29.05
1	LA01	B100	Base assembly	1	1	Each	16.35
2	B100	1100	Finished shaft	1	1	Each	1.30
3	1100	2100	3/8 Steel tubing	26	26	Inches	0.05
2	B100	1200	6-Diameter steel plate	1	1	Each	9.25
2	B100	1300	Hub	1	1	Each	5.00
2	B100	1400	1/4-20 Screw	4	4	Each	0.20
1	LA01	S100	Black shade	1	1	Each	4.10
1	LA01	A100	Socket assembly	1	1	Each	8.60
2	A100	1500	Steel holder	1	1	Each	0.40
3	1500	1400	1/4-20 Screw	2	2	Each	0.20
2	A100	1600	One-way socket	1	1	Each	3.50
2	A100	1700	Wiring assembly	1	1	Each	4.70
3	1700	2200	16-Gauge lamp cord	12	12	Feet	0.35
3	1700	2300	Standard plug terminal	1	1	Each	0.50

Example 3.9: Bill of Material Explosion

Example 3.8 illustrated using the Indented BOM data set to roll up costs assigned at the leaf nodes of the BOM family tree to obtain the cost of the final product. Similarly, the same data set can also be used to "explode" (or "propagate") requirements along the tree, based on requirements specified for the root nodes. The "explosion" can be performed with or without taking into account the quantities on hand or the scrap factors. This example demonstrates a few of these explosions.

Note that the BOM procedure automatically performs the requirement explosion in the calculation of the gross and net requirements of each item when producing the summarized parts list. In the Summarized Parts data set produced by PROC BOM, the gross and net requirements are calculated taking into account both the quantities on hand and the scrap factors. See the section "Summarized Parts Data Set" on page 37 for details about calculating the gross and net requirements for each item.

The following SAS code uses the data set SlBOM2 (displayed in Output 3.2.1) with additional specifications of scrap factors (20 percent for the relationship between the parent item '1100' and its component '2100', and 10 percent for the relationship of the component '2200' and its parent '1700') to produce the indented bill of material for 'LA01' and the summarized parts list for the requirement of 50 units of item 'LA01'. The indented BOM, IndBom9, is shown in Output 3.9.1, and the summarized parts list, SumBOM9, is shown in Output 3.9.2.

```
/* Product structure and part master data */
data SlBOM9;
set SlBOM2(drop=LeadTime);

/* Specify scrap factors for 2100 and 2200 */
if (Component="2100" and Parent="1100") then Scrap=0.2;
else if (Component="2200" and Parent="1700") then Scrap=0.1;
run;

  /* Create the indented BOM and the summarized parts list */
proc bom data=SlBOM9 out=IndBOM9 summaryout=SumBOM9;
   structure / part=Component
               parent=Parent
               component=Component
               quantity=QtyPer
               id=(Desc Unit)
               requirement=Gros_Req
               qtyonhand=On_Hand
               factor=Scrap;
run;
```

Output 3.9.1 Indented Bill of Material for 'LA01'

ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Desc	Qty Per	Qty_Prod	Unit	Scrap
0		LA01	Lamp LA	.	1	Each	.
1	LA01	B100	Base assembly	1	1	Each	0.0
2	B100	1100	Finished shaft	1	1	Each	0.0
3	1100	2100	3/8 Steel tubing	26	26	Inches	0.2
2	B100	1200	6-Diameter steel plate	1	1	Each	0.0
2	B100	1300	Hub	1	1	Each	0.0
2	B100	1400	1/4-20 Screw	4	4	Each	0.0
1	LA01	S100	Black shade	1	1	Each	0.0
1	LA01	A100	Socket assembly	1	1	Each	0.0
2	A100	1500	Steel holder	1	1	Each	0.0
3	1500	1400	1/4-20 Screw	2	2	Each	0.0
2	A100	1600	One-way socket	1	1	Each	0.0
2	A100	1700	Wiring assembly	1	1	Each	0.0
3	1700	2200	16-Gauge lamp cord	12	12	Feet	0.1
3	1700	2300	Standard plug terminal	1	1	Each	0.0

Output 3.9.2 Item Requirements for a Planned Order of 50 Units of 'LA01'

ABC Lamp Company

Summarized Parts List, Part LA01: Requirement=50

Part	Low_Code	Gros_Req	On_Hand	Net_Req	Desc	Unit
1100	2	0	0	0	Finished shaft	Each
1200	2	0	0	0	6-Diameter steel plate	Each
1300	2	0	0	0	Hub	Each
1400	3	60	0	60	1/4-20 Screw	Each
1500	2	30	0	30	Steel holder	Each
1600	2	30	0	30	One-way socket	Each
1700	2	30	0	30	Wiring assembly	Each
2100	3	0	0	0	3/8 Steel tubing	Inches
2200	3	396	0	396	16-Gauge lamp cord	Feet
2300	3	30	0	30	Standard plug terminal	Each
A100	1	30	0	30	Socket assembly	Each
B100	1	30	50	0	Base assembly	Each
LA01	0	50	20	30	Lamp LA	Each
S100	1	30	0	30	Black shade	Each

The summarized parts list displayed in Output 3.9.2 lists all items and their quantities required to be made or ordered in order to fill the requirement of 50 units for 'LA01'. The requirements in this list are calculated taking into account each item's quantity on hand and each relationship's scrap factor. However, if you want to analyze how future orders for end items will impact inventory levels, a *gross requirements report* that lists all items and their total requirements to make the pre-specified amount of end items, without any regard for quantities on hand, will be helpful. The gross requirements report can be determined by a top-down calculation through indented bill of material, taking into account only the scrap factors. In other words, once you have an Indented BOM data set available, you can use it to calculate the gross requirements report for any specified quantity of any end item, as shown below. It is not necessary to invoke the BOM procedure for each value.

The following code performs the top-down calculation, as discussed above, through the Indented BOM data set as displayed in Output 3.9.1 to produce a gross requirements report for an additional order of 50 units of 'LA01'. The gross requirements report, SumReq9, is displayed in Output 3.9.3; this data set has been sorted by the _Part_ variable.

```
/* Explode the gross requirement to low-level components */
data IndBOM9a;
set IndBOM9;
array reqs[4] req1 req2 req3 req4;
retain req1 req2 req3 req4 0;
drop req1 req2 req3 req4;

/* Calculate the gross requirement */
if _Level_=0 then Req=50;
else Req=reqs[_Level_]*QtyPer*(1.0+Scrap);

/* keep the requirement of the current level */
reqs[_Level_+1]=Req;
```

```
output;
run;

  /* Calculate the total requirement for each item */
proc sql;
   create table SumReq9 as
      select _Part_, Desc, Unit,
             sum(Req) as Gros_Req
      from IndBOM9a
      group by  _Part_, Desc, Unit;
quit;
```

Output 3.9.3 Total Requirements for Ordering 50 Units of 'LA01'

ABC Lamp Company

Gross Requirements Report, Part LA01: Requirement=50

Part	Desc	Unit	Gros_Req
1100	Finished shaft	Each	50
1200	6-Diameter steel plate	Each	50
1300	Hub	Each	50
1400	1/4-20 Screw	Each	300
1500	Steel holder	Each	50
1600	One-way socket	Each	50
1700	Wiring assembly	Each	50
2100	3/8 Steel tubing	Inches	1560
2200	16-Gauge lamp cord	Feet	660
2300	Standard plug terminal	Each	50
A100	Socket assembly	Each	50
B100	Base assembly	Each	50
LA01	Lamp LA	Each	50
S100	Black shade	Each	50

In another situation, you may like to know the quantity of each component used in making a certain amount of the end item, without any regard for items on hand and scrap factors. A similar calculation as described above can be used to determine the total usage for each item. In fact, the quantities of lower-level components that are used to make 1 unit of the end item are already contained in the variable Qty_Prod of the Indented BOM data set. You only need to multiply those quantities by the specified amount of the end item and then add the values for the same item together for each one. The following codes performs this task to create a report of the total usage for each item in making 50 units of 'LA01'. This usage report, SumUse9, has been sorted by the _Part_ variable and is shown in Output 3.9.4.

```
  /* Calculate the total usage for each item */
proc sql;
   create table SumUse9 as
      select _Part_, Desc, Unit,
             sum(Qty_Prod * 50) as Qty_Use
      from IndBOM9
      group by  _Part_, Desc, Unit;
quit;
```

Output 3.9.4 Total Usages for Making 50 Units of 'LA01'

ABC Lamp Company

Summarized Bill of Material, Part LA01: Requirment=50

Part	Desc	Unit	Qty_Use
1100	Finished shaft	Each	50
1200	6-Diameter steel plate	Each	50
1300	Hub	Each	50
1400	1/4-20 Screw	Each	300
1500	Steel holder	Each	50
1600	One-way socket	Each	50
1700	Wiring assembly	Each	50
2100	3/8 Steel tubing	Inches	1300
2200	16-Gauge lamp cord	Feet	600
2300	Standard plug terminal	Each	50
A100	Socket assembly	Each	50
B100	Base assembly	Each	50
LA01	Lamp LA	Each	50
S100	Black shade	Each	50

As discussed in Example 3.7, you can also use the %BOMRSUB SAS macro described in Chapter 4, "Bill of Material Postprocessing Macros," to create the same report as that shown in Output 3.9.4.

Example 3.10: Aggregating Forecasts Using PROC BOM

In this example, the BOM procedure is used in a nonstandard application. Recall that the input data for the procedure requires items that are related in a parent-child hierarchy. The procedure uses this information to create an output data set that lists the items in a top-down order that can be used to aggregate information in a variety of ways. Suppose, for example, that the ABC Lamp Company has forecasts that are available for lamps to be stocked at various stores. The stores are served by warehouses that are, in turn, served by distribution centers. In other words, the different units of business are organized as a "tree." If the company wishes to aggregate the forecasts to calculate the overall forecast at the distribution center, it can do so using the BOM procedure as follows.

```
/* The input data set */
data demand10;
format parent $14. child $14. code $2. ;
input parent & child & code $ pct nitems;
datalines;
Dist.Center 1  Warehouse 1    DC   1    .
Dist.Center 1  Warehouse 2    DC   1    .
Dist.Center 2  Warehouse 3    DC   1    .
Dist.Center 2  Warehouse 4    DC   1    .
Dist.Center 2  Warehouse 5    DC   1    .
Warehouse 1    Store 01       WH   1    .
Warehouse 1    Store 02       WH   1    .
Warehouse 1    Store 03       WH   1    .
```

```
Warehouse 2    Store 04       WH   1    .
Warehouse 2    Store 05       WH   1    .
Warehouse 3    Store 06       WH   1    .
Warehouse 3    Store 07       WH   1    .
Warehouse 3    Store 08       WH   1    .
Warehouse 4    Store 09       WH   1    .
Warehouse 4    Store 10       WH   .5   .
Warehouse 5    Store 11       WH   1    .
Warehouse 5    Store 10       WH   .5   .
Store 01       .              ST   .   10
Store 02       .              ST   .   20
Store 03       .              ST   .   10
Store 04       .              ST   .   20
Store 05       .              ST   .   10
Store 06       .              ST   .   20
Store 07       .              ST   .   10
Store 08       .              ST   .   20
Store 09       .              ST   .   10
Store 10       .              ST   .   20
Store 11       .              ST   .   10
;
```

The data set demand10 defines the parent-child relationships of the company's business units. It also specifies the forecasts, via the nitems variable, for each store. The code variable specifies the type of the business unit identified by the parent variable: '`DC`' for distribution center, '`WH`' for warehouse, and '`ST`' for store. Note that some business units may be stocked from more than one parent unit. The pct variable specifies the percentage of the items in the unit identified by the child variable that are from the parent unit. For example, '`Store 10`' is stocked evenly (50 percent each) from '`Warehouse 4`' and '`Warehouse 5`'.

The following code constructs the hierarchical structure of the ABC Lamp Company and aggregates the forecasts for stores up to each distribution center. The aggregated forecasts for stores, warehouses, and distribution centers are displayed in Output 3.10.1, Output 3.10.2, and Output 3.10.3, respectively.

```
  /* Create the hierarchy structure */
proc bom data=demand10 out=indbom10;
   structure / part=parent
               parent=parent
               component=child
               quantity=pct
               id=(nitems code);
run;

  /* Sort the output data set in reverse order */
proc sort data=indbom10;
   by descending part_id;
run;

  /* Aggregate forecasts through the hierarchy structure */
data accumulate10;
   set indbom10;
   array items[3] dcitems whitems stitems;
   retain dcitems whitems stitems 0;

   do i=1 to _level_;
```

```
      if nitems then items[i]=items[i] + pct * nitems;
   end;

   if nitems then items[_level_+1]=items[_level_+1] + nitems;

   output;

   do i=_level_+1 to 3;
     items[i]=0;
   end;
run;
  /* Display the forecast for each store */
proc sort data=accumulate10(where=(code="ST"))
          out=stores10
          nodupkey;
   by _part_;
run;

proc print data=stores10 noobs;
   title1 'ABC Lamp Company';
   title3 'Totals for Stores';
   var _Part_ code stitems;
run;
  /* Display the aggregated forecast for each warehouse */
proc sort data=accumulate10(where=(code="WH"))
          out=houses10
          nodupkey;
   by _part_;
run;

proc print data=houses10 noobs;
   title1 'ABC Lamp Company';
   title3 'Totals for Warehouses';
   var _Part_ code whitems;
run;
  /* Display the aggregated forecast for each distribution center */
proc sort data=accumulate10(where=(code="DC"))
          out=dcs10
          nodupkey;
   by _part_;
run;

proc print data=dcs10 noobs;
   title1 'ABC Lamp Company';
   title3 'Totals for Distribution Centers';
   format _Part_ $14. ;
   var _Part_ code dcitems;
run;
```

Output 3.10.1 Forecasts for Stores

ABC Lamp Company

Totals for Stores

Part	code	stitems
Store 01	ST	10
Store 02	ST	20
Store 03	ST	10
Store 04	ST	20
Store 05	ST	10
Store 06	ST	20
Store 07	ST	10
Store 08	ST	20
Store 09	ST	10
Store 10	ST	20
Store 11	ST	10

Output 3.10.2 Aggregated Forecasts for Warehouses

ABC Lamp Company

Totals for Warehouses

Part	code	whitems
Warehouse 1	WH	40
Warehouse 2	WH	30
Warehouse 3	WH	50
Warehouse 4	WH	20
Warehouse 5	WH	20

Output 3.10.3 Aggregated Forecasts for Distribution Centers

ABC Lamp Company

Totals for Distribution Centers

Part	code	dcitems
Dist.Center 1	DC	70
Dist.Center 2	DC	90

Statement and Option Cross-Reference Tables

The following table references the options in the BOM procedure that are illustrated by the examples in this section. Note that the DATA=, DUPLICATE=, OUT=, PMDATA=, and SUMMARYOUT= options are specified on the PROC BOM statement. All other options are specified on the STRUCTURE statement.

Table 3.7 Options Specified in Examples

Option	1	2	3	4	5	6	7	8	9	10
COMPONENT=	X	X	X	X	X	X	X	X	X	X
DATA=	X	X	X	X	X	X	X	X	X	X
DUPLICATE=						X				
ENDITEM=						X				
FACTOR=			X			X			X	
ID=	X	X	X	X	X	X	X	X	X	X
LEADTIME=		X	X	X	X	X	X			
OFFSET=			X			X				
OUT=	X	X	X	X	X	X	X	X	X	X
PARENT=	X	X	X	X	X	X	X	X	X	X
PART=	X	X	X	X	X	X	X	X	X	X
PMDATA=			X	X	X	X	X	X		
QTYONHAND=		X							X	
QUANTITY=	X	X	X	X	X	X	X	X	X	X
REQUIREMENT=		X		X			X		X	
RID=			X			X				
SUMMARYOUT=	X	X	X	X			X		X	

References

Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1983), *Data Structures and Algorithms*, Reading, MA: Addison-Wesley.

Clement, J., Coldrick, A., and Sari, J. (1992), *Manufacturing Data Structures: Building Foundations for Excellence with Bills of Materials and Process Information*, Essex Junction, VT: Oliver Wright Limited Publications.

Cox, J. F., III and Blackstone, J. H., Jr, eds. (1998), *APICS Dictionary*, Ninth Edition, Alexandria, VA: APICS.

Fogarty, D. W., Blackstone, J. H., and Hoffmann, T. R. (1991), *Production and Inventory Management*, Second Edition, Cincinnati, OH: South-Western Publishing.

Landvater, D. V. and Gray, C. D. (1989), *MRP II Standard System: A Handbook for Manufacturing Software Survival*, New York: John Wiley & Sons.

Plossl, G. (1995), *Orlicky's Material Requirements Planning*, Second Edition, New York: McGraw-Hill.

Chapter 4
Bill of Material Postprocessing Macros

Contents

Overview: BOM Postprocessing Macros

The bill of material postprocessing macros are divided into two groups based on their functionality. The reporting macros generate miscellaneous reports from the Indented BOM output data set of PROC BOM, or a data set that contains indented bill of material. The reports include:

- Single-level bill of material
- Single-level where-used list
- Indented bill of material
- Indented where-used list
- Summarized bill of material

- Summarized where-used list

See "Indented BOM Data Set" on page 32 in Chapter 3, "The BOM Procedure," for details on the Indented BOM data set.

The transactional macros perform specialized transactions for maintaining and updating the bill of material for a product, product line, plant, or company. The transactions include:

- Copy-and-paste
- Multiple-delete
- Multiple-replace
- Same-as-except

Used correctly, these transactions make it possible to avoid some clerical work and reduce the time to maintain the bill of material. However, if the transactions are used incorrectly, it may take hours to reconstruct what was changed automatically in a matter of seconds (Landvater and Gray 1989).

Input Data Sets

All SAS macros described in this chapter use a SAS input data set that contains the indented bill of material for all end items of the product line, plant, or company. This data set is referred to as the Indented BOM data set. It is typically the Indented BOM output data set created by PROC BOM. See "Indented BOM Data Set" on page 32 in Chapter 3 for details about the variables and special structure of this SAS data set. Not every variable in the Indented BOM data set is needed in all macros. The variables required by each macro depend on the requirements of the macro, and are listed in the corresponding section describing each macro. Note that the macros depend heavily on the form of the Indented BOM data set that

- contains the complete indented bill of material for each end item.
- lists the components of each indented bill of material in the *depth-first* order.

Refer to Aho, Hopcroft, and Ullman (1983) for details about depth-first ordering. See also "Indented BOM Data Set" on page 32 in Chapter 3 for details regarding the records in this data set.

A simple Indented BOM data set, IndBOM, is displayed in Figure 4.1. It contains two indented bill of material, one for the item '`A101`' and the other for the item '`LA01`'. You can easily check that each indented bill of material in this data set lists its components in depth-first order. The data set IndBOM is used as the Indented BOM input data set in all examples in this chapter.

You specify the name of the Indented BOM data set using the IN= parameter. If you do not provide a name when calling a postprocessing macro, the macro uses the most recently created SAS data set as the Indented BOM data set.

One SAS macro, %BOMTCNP, requires an additional data set that contains the product structure record for the new parent-component relationship that will be created while performing the copy-and-paste transaction.

This data set is referred to as the Product Structure data set. See "Product Structure Data Set" on page 30 in Chapter 3 for details about this SAS data set.

Another SAS macro, %BOMTSAE, requires another data set that contains the part master record for the item specified in the ROOT= parameter of the macro. This data set is referred to as the Part Master data set. See "Part Master Data Set" on page 29 in Chapter 3 for details about this SAS data set.

Figure 4.1 Indented BOM Data Set (IndBOM)

ABC Lamp Company

Indented Bill of Material

Obs	_Level_	_Parent_	Paren_ID	_Part_	Part_ID	Desc	Qty Per	Fscrap
1	0		.	A101	0	Three-way socket assem.	.	.
2	1	A101	0	1500	1	Steel holder	1	0.0
3	2	1500	1	1400	2	1/4-20 Screw	2	0.0
4	1	A101	0	1601	3	Three-way socket	1	0.0
5	1	A101	0	1700	4	Wiring assembly	1	0.0
6	2	1700	4	2200	5	16-Gauge lamp cord	12	0.1
7	2	1700	4	2210	6	14-Gauge lamp cord	12	0.1
8	2	1700	4	2300	7	Standard plug terminal	1	0.0
9	0		.	LA01	8	Lamp LA	.	.
10	1	LA01	8	A100	9	One-way socket assem.	1	0.0
11	2	A100	9	1500	10	Steel holder	1	0.0
12	3	1500	10	1400	11	1/4-20 Screw	2	0.0
13	2	A100	9	1600	12	One-way socket	1	0.0
14	2	A100	9	1700	13	Wiring assembly	1	0.0

Obs	Qty_Prod	Unit	Lead Time	Tot_Lead	LTOff	Tot_Off	SDate	EDate	_Prod_
1	1	Each	1	1	.	0	.	.	A101
2	1	Each	2	3	0	0	.	.	A101
3	2	Each	1	4	0	0	.	.	A101
4	1	Each	2	3	0	0	.	.	A101
5	1	Each	1	2	0	0	.	.	A101
6	12	Feet	2	4	0	0	.	07APR2001	A101
7	12	Feet	2	4	0	0	08APR2001	.	A101
8	1	Each	1	3	0	0	.	.	A101
9	1	Each	2	2	.	0	.	.	LA01
10	1	Each	1	3	2	2	.	.	LA01
11	1	Each	2	5	0	2	.	.	LA01
12	2	Each	1	6	0	2	.	.	LA01
13	1	Each	2	5	0	2	.	.	LA01
14	1	Each	1	4	0	2	.	.	LA01

Figure 4.1 *continued*

ABC Lamp Company

Indented Bill of Material

Obs	_Level_	_Parent_	Paren_ID	_Part_	Part_ID	Desc	Qty Per	Fscrap
15	3	1700	13	2200	14	16-Gauge lamp cord	12	0.1
16	3	1700	13	2210	15	14-Gauge lamp cord	12	0.1
17	3	1700	13	2300	16	Standard plug terminal	1	0.0
18	1	LA01	8	B100	17	Base assembly	1	0.0
19	2	B100	17	1100	18	Finished shaft	1	0.0
20	3	1100	18	2100	19	3/8 Steel tubing	26	0.2
21	2	B100	17	1200	20	6-Diameter steel plate	1	0.0
22	2	B100	17	1300	21	Hub	1	0.0
23	2	B100	17	1400	22	1/4-20 Screw	4	0.0
24	1	LA01	8	S100	23	Black shade	1	0.0

Obs	Qty_Prod	Unit	Lead Time	Tot_Lead	LTOff	Tot_Off	SDate	EDate	_Prod_
15	12	Feet	2	6	0	2	.	07APR2001	LA01
16	12	Feet	2	6	0	2	08APR2001	.	LA01
17	1	Each	1	5	0	2	.	.	LA01
18	1	Each	1	3	0	0	.	.	LA01
19	1	Each	2	5	0	0	.	.	LA01
20	26	Inches	3	8	0	0	.	.	LA01
21	1	Each	3	6	0	0	.	.	LA01
22	1	Each	2	5	1	1	.	.	LA01
23	4	Each	1	4	3	3	.	.	LA01
24	1	Each	2	4	0	0	.	.	LA01

Software Requirements

This collection of SAS macros uses procedures from Base SAS and SAS/OR software, as shown in the following table:

SAS Software Module	Procedure	Used in Macros
Base SAS	SORT	%BOMRSLB, %BOMRSLW, %BOMRSUB, %BOMRSUW, %BOMTCNP, %BOMTREP, %BOMTSAE
Base SAS	SQL	%BOMRSLW, %BOMRSUB, %BOMRSUW
SAS/OR	BOM	%BOMTCNP, %BOMTREP, %BOMTSAE

Calling the Macros

The macros are part of the SAS Autocall library, and are automatically available for use in your SAS program, provided the SAS system option MAUTOSOURCE is in effect. For more information about autocall libraries, refer to *SAS Macro Language: Reference*.

All macros are defined using *keyword parameters*. With keyword parameters, you do not have to keep track of the positions of the parameters when calling any of the bill of material postprocessing macros. The general format of a call to a macro in this chapter is:

```
%bommacro(keyword1=value1, keyword2=value2,..., keywordn=valuen);
```

Each keyword parameter allows you to initialize a macro variable with the same name as the parameter inside the macro program to the value after the equals sign of a parameter. It is not necessary to specify all the keyword parameters in a macro program call. If you do not specify a keyword parameter, the macro program uses the default value of the parameter to initialize the corresponding macro variable. If the parameter does not have a default value defined for it, the corresponding macro variable is initialized with a null value. In a macro program call, no value after the equals sign of a parameter initializes the corresponding macro variable with a null value. Refer to Chapter 4, "Macro Programs," in *SAS Macro Programming Made Easy* (Burlew 1998) for details about macros with keyword parameters.

Output Data Sets

Each macro in this chapter creates a SAS output data set. You can provide the name of this data set by specifying the OUT= parameter in each macro. If you do not provide a name when calling a postprocessing macro, the macro creates an output data set named _BOMOUT_. This data set contains the same variables as the Indented BOM data set, except for those variables that are specified in the DROP= parameter. Some macros add new variables to the output data sets for particular information, such as quantity used and quantity needed, that are not contained in the Indented BOM data set.

Macro Variables

Each bill of material postprocessing macro defines a macro variable named *_MACRONAME_*, where "*MACRONAME*" is the name of the macro. Each macro variable contains a character string that indicates the status of the corresponding macro execution. It is set at the completion of the macro execution. The form of the character string is STATUS=*status*, where *status* can be one of the following:

SUCCESSFUL	indicates successful completion of the macro.
IO_ERROR	indicates that the macro failed to open the input data sets or to create the output data set.
RUNTIME_ERROR	indicates that macro execution was halted due to an error found by the SAS System.
RUNTIME_WARNING	indicates that macro execution was completed, but a warning was issued by the SAS System.

SYNTAX_ERROR	indicates that macro execution was halted due to an error in the macro invocation syntax.
SYNTAX_WARNING	indicates that macro execution was completed, but a warning in the macro invocation syntax was issued.

You can check the SAS log for detailed error or warning messages if a macro variable has a value other than "STATUS=SUCCESSFUL". These messages will help you determine the source of the errors. You can also use MPRINT, MLOGIC, SYMBOLGEN, or other SAS System options to help track down errors. For details, refer to the "Macro Facility Error Messages and Debugging" chapter in *SAS Macro Language: Reference*.

Reporting Macros

The reporting macros are used to construct reports from an Indented BOM output data set as produced by the BOM procedure. The macros contained in this section are as follows:

%BOMRMLB	constructs an indented bill of material
%BOMRMLW	constructs an indented where-used list
%BOMRSLB	constructs a single-level bill of material
%BOMRSLW	constructs a single-level where-used list
%BOMRSUB	constructs a summarized bill of material
%BOMRSUW	constructs a summarized where-used list

The *single-level bill of material* lists the components directly used by a given item. The *indented bill of material* of a given item lists all components that are directly or indirectly used by that item. It starts with the single-level bill of material for the given item. Each component in this bill of material is checked to see if it has subcomponents or a single-level bill of material. If so, the component is exploded. If any component on this second level bill has subcomponents, that component is exploded in the same fashion. The explosion of bill of material continues until the lowest levels are reached (Landvater and Gray 1989). The *summarized bill of material* lists all components and their quantities that are directly or indirectly used to make prespecified units of the given item.

On the other hand, the *single-level where-used list* traces all parent items where the given item is directly used. The *indented where-used list* starts with the single-level where-used list of the given item, and traces upward to all the places each parent item is used, and so on until the top-level end item is reached. The *summarized where-used list* shows all parents at higher levels in which a given item is directly or indirectly used, along with the total quantities of the item used in those parents.

%BOMRMLB: Indented Bill of Material

%bomrmlb (*ROOT=root, IN=SAS-data-set, LEV=variable, PARENT=variable,*
COMP=variable, QUANTITY=variable, QTYPROD=variable,
FACTOR=variable, LEADTIME=variable, TOTLEAD=variable,
OFFSET=variable, TOTOFF=variable, PARENID=variable,
PROD=variable, REQ=req, OUT=SAS-data-set, DROP=variables);

constructs an indented bill of material for a given item referred to as the item Root, which lists all components directly or indirectly used in the item Root. If a component is a subassembly, all its components and all their components are also listed, down to purchased items and raw materials. The parameters for this macro are as follows:

ROOT=*root*
specifies the part number of the item for which the indented bill of material is constructed. This item is referred to as the item Root. You must specify either a number or a quoted string for the ROOT= parameter, depending on the type of the Component variable.

IN=*SAS-data-set*
names the Indented BOM input data set. The default value is the most recently created SAS data set. A null value is replaced by the default value.

LEV=*variable*
identifies the variable in the Indented BOM data set that contains the indenture level information. The default value is _Level_. A null value is replaced by the default value.

PARENT=*variable*
identifies the variable in the Indented BOM data set that contains the parent item's part number. A null value tells the macro program not to look for the variable in the input data set that contains the parent item's part number. The default value is _Parent_.

COMP=*variable*
identifies the variable in the Indented BOM data set that contains the component's part number. This variable is referred to as the Component variable. The default value is _Part_. A null value is replaced by the default value.

QUANTITY=*variable*
identifies the variable in the Indented BOM data set that contains the quantity per assembly information. A null value tells the macro program not to look for the variable in the input data set that contains the quantity per assembly information. The default value is Qty_Per.

QTYPROD=*variable*
identifies the variable in the Indented BOM data set that contains the quantity per product information. A null value tells the macro program not to look for the variable in the input data set that contains the quantity per product information. The default value is Qty_Prod.

FACTOR=*variable*
identifies the variable in the Indented BOM data set that contains the scrap factor information. A null value tells the macro program not to look for the variable in the input data set that contains the scrap factor information. The default value is S_Factor.

LEADTIME=*variable*
identifies the variable in the Indented BOM data set that contains the lead time information. If you do not specify this parameter or specify a null value, the macro program will not look for the variable in the input data set that contains the lead time information.

TOTLEAD=*variable*
identifies the variable in the Indented BOM data set that contains the total lead time information. A null value tells the macro program not to look for the variable in the input data set that contains the total lead time information. The default value is Tot_Lead. The specification of the TOTLEAD= parameter is ignored if the macro program cannot find the variable in the Indented BOM data set that contains the lead time information.

OFFSET=*variable*
identifies the variable in the Indented BOM data set that contains the lead-time offset information. A null value tells the macro program not to look for the variable in the input data set that contains the lead-time offset information. The default value is L_Offset.

TOTOFF=*variable*
identifies the variable in the Indented BOM data set that contains the total lead-time offset information. A null value tells the macro program not to look for the variable in the input data set that contains the total lead-time offset information. The default value is Tot_Off. The specification of the TOTOFF= parameter is ignored if the macro program cannot find the variable in the Indented BOM data set that contains the lead-time offset information.

PARENID=*variable*
identifies the variable in the Indented BOM data set that contains the ID number for the parent record. A null value tells the macro program not to look for the variable in the input data set that contains the ID number of the parent record. The default value is Paren_ID.

PROD=*variable*
identifies the variable in the Indented BOM data set that contains the end item's part number. A null value tells the macro program not to look for the variable in the input data set that contains the end item's part number. The default value is _Prod_.

REQ=*req*
specifies the quantity required for the item Root. The default value is 1. A null value is replaced by the default value.

OUT=*SAS-data-set*
names the output data set that contains indented bill of material for the item Root. The default value is _BOMOUT_. A null value is replaced by the default value.

DROP=*variables*
identifies all the variables in the Indented BOM data set that you do not want to be included in the output data set. You can list the variables in any form that SAS allows. If you do not specify the DROP= parameter, or specify a null value, all variables in the Indented BOM data set are included in the output data set.

The Input Data Set

The input data set for this macro is the Indented BOM data set described in "Input Data Sets" on page 82.

The following parameters specify the variables in the input data set that are required by the %BOMRMLB macro.

COMP= A character or numeric variable that contains the part number of the component.

LEV= A numeric variable that contains the indenture level information.

The following parameters specify the variables in the Indented BOM input data set that are used by the macro, if they can be found in the data set.

FACTOR= A numeric variable that contains the scrap factor information.

LEADTIME= A numeric variable that contains the lead time of the component.

OFFSET= A numeric variable that contains the lead-time offset information.

PARENID= A numeric variable that contains the ID number for the parent record.

PARENT= A character or numeric variable that contains the part number of the parent item. The type of this variable must be the same as that of the variable specified in the COMP= parameter.

PROD= A character or numeric variable that contains the part number of the end item. The type of this variable must be the same as that of the variable specified in the COMP= parameter.

QTYPROD= A numeric variable that contains the quantity per product information.

QUANTITY= A numeric variable that contains the quantity per assembly information.

TOTLEAD= A numeric variable that contains the total lead time information.

TOTOFF= A numeric variable that contains the total lead-time offset information.

DROP= Variables that you do not want to be included in the output data set.

The Output Data Set

The output data set contains the indented bill of material for the item Root. This data set is a subset of the Indented BOM input data set, and has the same variables as the Indented BOM data set, except for the variables that are specified in the DROP= parameter. The quantity per product (the value of the variable specified in the QTYPROD= parameter) of each record in this data set reflects the quantity needed in producing *m* units of the item Root, where *m* is the value specified in the REQ= parameter.

Note that the first record of the output data set does not contain a parent-component relationship. Therefore, the values for all variables that contain product structure information in this record have to be missing. The %BOMRMLB macro sets the values for the variables that are specified in the PARENT=, QUANTITY=, FACTOR=, and OFFSET= parameters to a missing value. It also sets the value of the variable specified in the PARENID= parameter to a missing value since the first record does not have a parent record. If there are other variables that also contain product structure data, you must manually set the values for those variables in the first record. See "Product Structure Data Set" on page 30 in Chapter 3 for the variables that contain product structure information.

Macro Variable _BOMRMLB_

Upon completion, this macro defines a macro variable, _BOMRMLB_, which contains a character string that indicates the status of the macro execution. See "Macro Variables" on page 85 for the possible values of this macro variable.

An Example

The following statement constructs an indented bill of material for the item 'A100' from the Indented BOM data set displayed in Figure 4.1.

```
%bomrmlb(root='A100', in=IndBOM, quantity=QtyPer,
         factor=Fscrap, leadtime=LeadTime, offset=LTOff,
         out=BomOut1, drop=Paren_ID Part_ID _Prod_);
```

The indented bill of material for the item 'A100' is displayed in Figure 4.2.

Figure 4.2 Indented Bill of Material for Item A100 (BomOut1)

ABC Lamp Company
%BOMRMLB macro
Indented Bill of Material, Part A100

_Level	_Parent	Part	Description	Qty	Scrap	Qty_Prod	Unit	LTime	LTOff	_Prod_	StartDate	EndDate
0		A100	One-way socket assem.	.	.	1	Each	1	1	. 0	.	.
1	A100	1500	Steel holder	1	0.0	1	Each	2	3	0 0	.	.
2	1500	1400	1/4-20 Screw	2	0.0	2	Each	1	4	0 0	.	.
1	A100	1600	One-way socket	1	0.0	1	Each	2	3	0 0	.	.
1	A100	1700	Wiring assembly	1	0.0	1	Each	1	2	0 0	.	.
2	1700	2200	16-Gauge lamp cord	12	0.1	12	Feet	2	4	0 0	.	07APR2001
2	1700	2210	14-Gauge lamp cord	12	0.1	12	Feet	2	4	0 0	08APR2001	.
2	1700	2300	Standard plug terminal	1	0.0	1	Each	1	3	0 0	.	.

%BOMRMLW: Indented Where-Used List

%bomrmlw (*ROOT=root, IN=SAS-data-set, LEV=variable, COMP=variable, QTYPROD=variable, QTYUSED=qtyused, OUT=SAS-data-set, DROP=variables*);

constructs an indented where-used list for a given item referred to as the item Root, which lists every parent item of the item Root and the respective quantities required, as well as each of their respective parent items, continuing until the end item is referenced (Cox and Blackstone 1998). The parameters for this macro are as follows:

ROOT=*root*

specifies the part number of the item for which the indented where-used list is constructed. This item is referred to as the item Root. You must specify either a number or a quoted string for the ROOT= parameter, depending on the type of the Component variable.

IN=*SAS-data-set*
names the Indented BOM input data set. The default value is the most recently created SAS data set. A null value is replaced by the default value.

LEV=*variable*
identifies the variable in the Indented BOM data set that contains the indenture level information. The default value is _Level_. A null value is replaced by the default value.

COMP=*variable*
identifies the variable in the Indented BOM data set that contains the component's part number. This variable is also referred to as the Component variable. The default value is _Part_. A null value is replaced by the default value.

QTYPROD=*variable*
identifies the variable in the Indented BOM data set that contains the quantity per product information. The default value is Qty_Prod. A null value tells the macro program not to look for the variable in the input data set that contains the quantity per product information.

QTYUSED=*qtyused*
specifies the name for the variable in the output data set that contains the quantity of the item Root used by the item identified by the Component variable. A null value tells the macro program not to calculate and write the quantity-used information to the output data set. The default value is Qty_Used. The specification of the QTYUSED= parameter is ignored if the macro program cannot find the variable in the Indented BOM data set that contains the quantity per product information. The Indented BOM data set should not contain a variable with the name specified in the QTYUSED= parameter. Otherwise, the values of the variable with the name specified in the QTYUSED= parameter will be overwritten.

OUT=*SAS-data-set*
names the output data set that contains indented where-used list data for the item Root. The default value is _BOMOUT_. The null value is replaced by the default value.

DROP=*variables*
specifies all *variables* in the Indented BOM data set that you do not want to be included in the output data set. You can list the variables in any form that SAS allows. The default value is the variable that is specified in the QTYPROD= parameter. A null value tells the macro program to write all the variables in the Indented BOM data set to the output data set.

The Input Data Set

The input data set of this macro is the Indented BOM data set described in "Input Data Sets" on page 82.

The following parameters specify the variables in the input data set that are required by the %BOMRMLW macro.

COMP=	A character or numeric variable that contains the part number of the component.
LEV=	A numeric variable that contains the indenture level information.

The following parameters specify the variables in the Indented BOM input data set that are used by the macro, if they can be found in the data set.

QTYPROD= A numeric variable that contains the quantity per product information.

DROP= Variables that you do not want to be included in the output data set.

The Output Data Set

The output data set contains the indented where-used list for the item Root. This data set has the same variables as the Indented BOM data set, except for those variables that are specified in the DROP= parameter. If both values for QTYPROD= and QTYUSED= parameters are not null, the output data set contains a new variable with the name specified by the following parameter:

QTYUSED= A numeric variable that contains the quantity of the item Root that is used in the item identified by the Component variable.

Macro Variable _BOMRMLW_

Upon completion, this macro defines a macro variable, _BOMRMLW_, which contains a character string that indicates the status of the macro execution. See "Macro Variables" on page 85 for the possible values of this macro variable.

An Example

The following statement constructs an indented where-used list for the item '1400' from the Indented BOM data set displayed in Figure 4.1.

```
%bomrmlw(root='1400', in=IndBOM, qtyused=QtyUsed,
out=BomOut2, drop=Paren_ID Part_ID Qty_Prod);
```

The indented where-used list for the item '1400' is displayed in Figure 4.3. In this list, the variable QtyUsed contains the quantity of the item '1400' that is used in the item identified by the variable _Part_. For example, if you trace back from the item '1400' up to the end item 'A101', you will observe that the item '1500' uses 2 units of '1400', and the item 'A101' uses 1 unit of '1500', and hence, 2 units of '1400'. Therefore, the value for the variable QtyUsed is 2 on both observations 2 and 3.

Figure 4.3 Indented Where-Used List for Item 1400 (BomOut2)

ABC Lamp Company
%BOMRMLW macro
Indented Where-Used List, Part 1400

Obs	_Level_	_Parent_	_Part_	Desc	Qty Per	Fscrap	Qty Used
1	2	1500	1400	1/4-20 Screw	2	0	.
2	1	A101	1500	Steel holder	1	0	2
3	0		A101	Three-way socket assem.	.	.	2
4	3	1500	1400	1/4-20 Screw	2	0	.
5	2	A100	1500	Steel holder	1	0	2
6	1	LA01	A100	One-way socket assem.	1	0	2
7	0		LA01	Lamp LA	.	.	2
8	2	B100	1400	1/4-20 Screw	4	0	.
9	1	LA01	B100	Base assembly	1	0	4
10	0		LA01	Lamp LA	.	.	4

Obs	Unit	Lead Time	Tot_Lead	LTOff	Tot_Off	SDate	EDate	_Prod_
1	Each	1	4	0	0	.	.	A101
2	Each	2	3	0	0	.	.	A101
3	Each	1	1	.	0	.	.	A101
4	Each	1	6	0	2	.	.	LA01
5	Each	2	5	0	2	.	.	LA01
6	Each	1	3	2	2	.	.	LA01
7	Each	2	2	.	0	.	.	LA01
8	Each	1	4	3	3	.	.	LA01
9	Each	1	3	0	0	.	.	LA01
10	Each	2	2	.	0	.	.	LA01

%BOMRSLB: Single-Level Bill of Material

%bomrslb (*ROOT=root, IN=SAS-data-set, PARENT=variable, COMP=variable, OUT=SAS-data-set, DROP=variables*);

constructs a single-level bill of material for a given item referred to as the item Root, which lists all components that are directly used in the item Root. The parameters for this macro are as follows:

ROOT=*root*

specifies the part number of the item for which the single-level bill of material is constructed. This item is referred to as the item Root. You must specify either a number or a quoted string for the ROOT= parameter, depending on the type of the Component variable.

IN=*SAS-data-set*

names the Indented BOM input data set. The default value is the most recently created SAS data set. A null value is replaced by the default value.

PARENT=*variable*
identifies the variable in the Indented BOM data set that contains the parent item's part number. The default value is _Parent_. A null value is replaced by the default value.

COMP=*variable*
identifies the variable in the Indented BOM data set that contains the component's part number. This variable is also referred to as the Component variable. The default value is _Part_. A null value is replaced by the default value.

OUT=*SAS-data-set*
names the output data set that contains single-level bill of material data for the item Root. The default value is _BOMOUT_. A null value is replaced by the default value.

DROP=*variables*
specifies all variables in the Indented BOM data set that you do not want to be included in the output data set. You can list the variables in any form that SAS allows. The default value is the list of variables _Level_, Paren_ID, Part_ID, _Prod_, Qty_Prod, Tot_Lead, and Tot_Off, if they are in the Indented BOM data set. A null value tells the macro program to write all variables in the Indented BOM data set to the output data set.

The Input Data Set

The input data set of this macro is the Indented BOM data set described in "Input Data Sets" on page 82.

The following parameters specify the variables in the input data set that are required by the %BOMRSLB macro.

COMP= A character or numeric variable that contains the part number of the component.

PARENT= A character or numeric variable that contains the part number of the parent item. The type of this variable must be the same as that of the variable specified in the COMP= parameter.

The following parameter specifies the variables in the Indented BOM input data set that are used by the macro, if they can be found in the data set.

DROP= Variables that you do not want to be included in the output data set.

The Output Data Set

The output data set contains the single-level bill of material for the item Root. This data set has the same variables as the Indented BOM data set, except for those variables that are specified in the DROP= parameter.

Macro Variable _BOMRSLB_

Upon completion, this macro defines a macro variable, _BOMRSLB_, which contains a character string that indicates the status of the macro execution. See "Macro Variables" on page 85 for the possible values of this macro variable.

An Example

The following statement constructs a single-level bill of material for the item '`B100`' from the Indented BOM data set displayed in Figure 4.1.

```
%bomrslb(root='B100', in=IndBOM, out=BomOut3);
```

The single-level bill of material for the item '`B100`' is displayed in Figure 4.4.

Figure 4.4 Single-Level Bill of Material for Item B100 (BomOut3)

ABC Lamp Company
%BOMRSLB macro
Single-Level Bill of Material, Part B100

_ Parent _	_ Part _	Description	Qty Per	Scrap Factor	Unit of Measure	Lead Time Offset	Level	Start Date	End Date
B100	1100	Finished shaft	1	0	Each	2	0	.	.
B100	1200	6-Diameter steel plate	1	0	Each	3	0	.	.
B100	1300	Hub	1	0	Each	2	1	.	.
B100	1400	1/4-20 Screw	4	0	Each	1	3	.	.

%BOMRSLW: Single-Level Where-Used List

%bomrslw (*ROOT=root, IN=SAS-data-set, PARENT=variable, COMP=variable, QUANTITY=variable, PARENID=variable, PARTID=variable, RID=variables, QTYUSED=qtyused, OUT=SAS-data-set, DROP=variables*);

constructs a single-level where-used list for a given item referred to as the item Root, which lists each parent item in which the item Root is directly used and in what quantity. The parameters for this macro are as follows:

ROOT=*root*

specifies the part number of the item for which the single-level where-used list is constructed. This item is referred to as the item Root. You must specify either a number or a quoted string for the ROOT= parameter, depending on the type of the Component variable.

IN=*SAS-data-set*

names the Indented BOM input data set. The default value is the most recently created SAS data set. A null value is replaced by the default value.

PARENT=*variable*
identifies the variable in the Indented BOM data set that contains the parent item's part number. The default value is _Parent_. A null value is replaced by the default value.

COMP=*variable*
identifies the variable in the Indented BOM data set that contains the component's part number. This variable is also referred to as the Component variable. The default value is _Part_. A null value of the corresponding macro variable is replaced by this default value in the macro program.

QUANTITY=*variable*
identifies the variable in the Indented BOM data set that contains the quantity per assembly information. A null value tells the macro program not to look for the variable in the input data set that contains the quantity per assembly information. The default value is Qty_Per.

PARENID=*variable*
identifies the variable in the Indented BOM data set that contains the ID number for the parent record. The default value is Paren_ID. A null value is replaced by the default value.

PARTID=*variable*
identifies the variable in the Indented BOM data set that contains the ID number for the current record. The default value is Part_ID. A null value is replaced by the default value.

RID=*variables*
identifies variables in the Indented BOM data set that contain the product structure information, and are not specified in PARENT=, COMP=, or QUANTITY= parameters. If you do not specify this parameter or specify a null value, the macro program assumes the Indented BOM data set contains no RID variables.

QTYUSED=*qtyused*
specifies the name for the variable in the output data set that contains the quantity of the item Root used by the parent item identified by the Component variable. A null value tells the macro program not to calculate and write the quantity-used information to the output data set. The default value is Qty_Used. The specification of the QTYUSED= parameter is ignored if the macro program cannot find the variable in the Indented BOM data set that contains the quantity per assembly information. The Indented BOM data set should not contain any variables with the name specified by the QTYUSED= parameter. Otherwise, the values of the variable with the name specified in the QTYUSED= parameter will be overwritten.

OUT=*SAS-data-set*
names the output data set that contains a single-level where-used list for the item Root. The default value is _BOMOUT_. A null value is replaced by this default value.

DROP=*variables*
specifies all variables in the Indented BOM data set that you do not want to be included in the output data set. You can list the variables in any form that SAS allows. The default value is the list that contains the variables specified in the COMP=, QUANTITY=, PARENID=, and PARTID= parameters; plus the _Level_, _Prod_, Qty_Prod, Tot_Lead, and Tot_Off variables if they are in the Indented BOM data set. A null value tells the macro program to write all the variables in the Indented BOM data set to the output data set.

The Input Data Set

The input data set of this macro is the Indented BOM data set described in "Input Data Sets" on page 82.

The following parameters specify the variables in the input data set that are required by the %BOMRSLW macro.

COMP= A character or numeric variable that contains the part number of the component.

PARENID= A numeric variable that contains the ID number for the parent record.

PARENT= A character or numeric variable that contains the part number of the parent item. The type of this variable must be the same as that of the variable specified in the COMP= parameter.

PARTID= A numeric variable that contains the ID number for the current record.

The following parameters specify the variables in the Indented BOM input data set that are used by the macro, if they can be found in the data set.

QUANTITY= A numeric variable that contains the quantity per assembly information.

RID= Variables that contain product structure information, except for those specified in the COMP=, PARENT=, or QUANTITY= parameters.

DROP= Variables that you do not want to be included in the output data set.

The Output Data Set

The output data set contains the single-level where-used list for the item Root. This data set has the same variables as the Indented BOM data set, except for the variables specified in the DROP= parameter. If both values for the QUANTITY= and QTYUSED= parameters are not null, the output data set contains a new variable with the name specified by the following parameter:

QTYUSED= A numeric variable that contains the quantity of the item Root that is used in the parent item identified by the Component variable.

Macro Variable _BOMRSLW_

Upon completion, this macro defines a macro variable, _BOMRSLW_, which contains a character string that indicates the status of the macro execution. See "Macro Variables" on page 85 for the possible values of this macro variable.

An Example

The following statement constructs a single-level where-used list for the item '1400' from the Indented BOM data set displayed in Figure 4.1.

```
%bomrslw(root='1400', in=IndBOM, quantity=QtyPer,
rid=SDate EDate, qtyused=QtyUsed, out=BomOut4);
```

The single-level where-used list for the item '1400' is displayed in Figure 4.5.

Figure 4.5 Single-Level Where-Used List for Item 1400 (BomOut4)

ABC Lamp Company
%BOMRSLW macro
Single-Level Where-Used List, Part 1400

Parent	Desc	Qty Used	Fscrap	Unit	Lead Time	LTOff	SDate	EDate
1500	Steel holder	2	0	Each	2	0	.	.
B100	Base assembly	4	0	Each	1	0	.	.

%BOMRSUB: Summarized Bill of Material

%bomrsub (*ROOT=root, IN=SAS-data-set, LEV=variable, PARENT=variable, COMP=variable, QUANTITY=variable, QTYPROD=variable, RID=variables, QTYREQ=qtyreq, REQ=req, OUT=SAS-data-set, DROP=variables*);

constructs a summarized bill of material for a given item referred to as the item Root, which lists all the components and their quantities used to make *m* units of the item Root, where *m* is the value specified in the REQ= parameter. Unlike the indented bill of material, it does not list the levels of manufacture and lists a component only once for the total quantity used (Cox and Blackstone 1998). Note that the summarized bill of material created by this macro is more general than the one described in Chapter 3, where the total quantity used of each component is for making 1 unit of the item Root. The form of the summarized bill of material constructed by this macro enables you to determine how a future order for *m* units of the item Root will impact inventory levels. The parameters for this macro are as follows:

ROOT=*root*

specifies the part number of the item for which the summarized bill of material is constructed. This item is referred to as the item Root. You must specify either a number or a quoted string for the ROOT= parameter, depending on the type of the Component variable.

IN=*SAS-data-set*

names the Indented BOM input data set. The default value is the most recently created SAS data set. A null value is replaced by the default value.

LEV=*variable*

identifies the variable in the Indented BOM data set that contains the indenture level information. The default value is _Level_. A null value is replaced by the default value.

PARENT=*variable*

identifies the variable in the Indented BOM data set that contains the parent item's part number. A null value tells the macro program not to look for the variable in the input data set that contains the parent item's part number. The default value is _Parent_.

COMP=*variable*
identifies the variable in the Indented BOM data set that contains the component's part number. This variable is referred to as the Component variable. The default value is _Part_. A null value is replaced by the default value.

QUANTITY=*variable*
identifies the variable in the Indented BOM data set that contains the quantity per assembly information. A null value tells the macro program not to look for the variable in the input data set that contains the quantity per assembly information. The default value is Qty_Per.

QTYPROD=*variable*
identifies the variable in the Indented BOM data set that contains the quantity per product information. The default value is Qty_Prod. A null value is replaced by the default value.

RID=*variables*
identifies variables in the Indented BOM data set that contain product structure information and are not specified in the PARENT=, COMP=, or QUANTITY= parameters. These variables are only needed in creating the default drop variables. If you do not specify this parameter or specify a null value, the macro program assumes the Indented BOM data set contains no RID variables.

QTYREQ=*qtyreq*
specifies the name for the variable in the output data set that contains the quantity of the item identified by the Component variable that is used to make *m* units of the item Root, where *m* is the value specified in the REQ= parameter. The default value is Qty_Req. A null value is replaced by the default value. The Indented BOM data set should not contain any variables with the name specified by the QTYREQ= parameter. Otherwise, the values of the variable with the name specified in the QTYREQ= parameter will be overwritten.

REQ=*req*
specifies the quantity required for the item Root. The default value is 1. A null value is replaced by the default value.

OUT=*SAS-data-set*
names the output data set that contains summarized bill of material data. The default value is _BOMOUT_. A null value is replaced by this default value.

DROP=*variables*
identifies all variables in the Indented BOM data set that you do not want to be included in the output data set. You can list the variables in any form that SAS allows. The default value is the list that contains the variables specified in the LEV=, PARENT=, QUANTITY=, QTYPROD=, and RID= parameters, in addition to the Paren_ID, Part_ID, _Prod_, Tot_Lead, and Tot_Off variables, if they are in the Indented BOM data set. A null value tells the macro program to write all the variables in the Indented BOM data set to the output data set.

The Input Data Set

The input data set of this macro is the Indented BOM data set described in "Input Data Sets" on page 82.

The following parameters specify the variables in the input data set that are required by the %BOMRSUB macro.

COMP= A character or numeric variable that contains the part number of the component.

LEV= A numeric variable that contains the indenture level information.

QTYPROD= A numeric variable that contains the quantity per product information.

The following parameters specify the variables in the Indented BOM input data set that are used by the macro, if they can be found in the data set.

PARENT= A character or numeric variable that contains the part number of the parent item. The type of this variable must be the same as that of the variable specified in the COMP= parameter.

QUANTITY= A numeric variable that contains the quantity per assembly information.

RID= Variables that contain product structure information, except for those specified in the PARENT=, COMP=, or QUANTITY= parameters. These variables are only needed in creating the default drop variables.

DROP= Variables that you do not want to be included in the output data set.

The Output Data Set

The output data set contains the summarized bill of material for the item Root. This data set has the same variables as the Indented BOM data set, except for those variables specified in the DROP= parameter. The output data set contains a new variable with the name specified by the following parameter:

QTYREQ= A numeric variable that contains the quantity of the component identified by the Component variable that is used to produce *m* units of the item Root, where *m* is the value specified in the REQ= parameter.

Macro Variable _BOMRSUB_

Upon completion, this macro defines a macro variable, _BOMRSUB_, which contains a character string that indicates the status of the macro execution. See "Macro Variables" on page 85 for the possible values of this macro variable.

An Example

The following statement constructs a summarized bill of material for the item '`LA01`' from the Indented BOM data set displayed in Figure 4.1.

```
%bomrsub(root='LA01', in=IndBOM, quantity=QtyPer, qtyreq=QtyReq,
rid=SDate EDate LTOff Fscrap, req=50, out=BomOut5);
```

The summarized bill of material for the item '`LA01`' is displayed in Figure 4.6. Note that a summarized bill of material lists a component only once for the total quantity used. For example, the total quantity of 300 units of the item '`1400`' comes from 200 units that are for making 50 units of '`B100`', and 100 units that are for producing 50 units of '`1500`'.

Figure 4.6 Summarized Bill of Material for Item LA01 (BomOut5)

ABC Lamp Company
%BOMRSUB macro
Summarized Bill of Material, Part LA01: Requirement=50

Part	Desc	Unit	Lead Time	Qty Req
1100	Finished shaft	Each	2	50
1200	6-Diameter steel plate	Each	3	50
1300	Hub	Each	2	50
1400	1/4-20 Screw	Each	1	300
1500	Steel holder	Each	2	50
1600	One-way socket	Each	2	50
1700	Wiring assembly	Each	1	50
2100	3/8 Steel tubing	Inches	3	1300
2200	16-Gauge lamp cord	Feet	2	600
2210	14-Gauge lamp cord	Feet	2	600
2300	Standard plug terminal	Each	1	50
A100	One-way socket assem.	Each	1	50
B100	Base assembly	Each	1	50
S100	Black shade	Each	2	50

%BOMRSUW: Summarized Where-Used List

%bomrsuw (*ROOT=root, IN=SAS-data-set, LEV=variable, COMP=variable, QUANTITY=variable, QTYPROD=variable, RID=variables, QTYUSED=qtyused, OUT=SAS-data-set, DROP=variables*);

constructs a summarized where-used list for a given item referred to as the item Root, which lists all parent items in which the item Root is directly or indirectly used and the required quantities. Unlike the indented where-used list, it does not list the levels of manufacture (Cox and Blackstone 1998). The parameters for this macro are as follows:

ROOT=*root*

specifies the part number of the item for which the summarized where-used list is constructed. This item is referred to as the item Root. You must specify either a number or a quoted string for the ROOT= parameter, depending on the type of the Component variable.

IN=*SAS-data-set*

names the Indented BOM input data set. The default value is the most recently created SAS data set. A null value is replaced by the default value.

LEV=*variable*

identifies the variable in the Indented BOM data set that contains the indenture level information. The default value is _Level_. A null value is replaced by the default value.

COMP=*variable*
identifies the variable in the Indented BOM data set that contains the component's part number. This variable is referred to as the Component variable. The default value is _Part_. A null value is replaced by the default value.

QUANTITY=*variable*
identifies the variable in the Indented BOM data set that contains the quantity per assembly information. This variable is only needed in creating the default drop variables. The default value is Qty_Per. A null value tells the macro program not to look for the variable that contains the quantity per assembly information.

QTYPROD=*variable*
identifies the variable in the Indented BOM data set that contains the quantity per product information. The default value is Qty_Prod. A null value is replaced by the default value.

RID=*variables*
identifies variables in the Indented BOM data set that contain product structure information and are not specified in COMP= or QUANTITY= parameters. These variables are only needed in creating the default drop variables. If you do not specify this parameter or specify a null value, the macro program assumes the Indented BOM data set contains no RID variables.

QTYUSED=*qtyused*
specifies the name for the variable in the output data set that contains the quantity of the item Root used by the parent item identified by the Component variable. The default value is Qty_Used. A null value is replaced by the default value. The Indented BOM data set should not contain any variables with the name specified by the QTYUSED= parameter. Otherwise, the values of the variable with the name specified in the QTYUSED= parameter will be overwritten.

OUT=*SAS-data-set*
names the output data set that contains summarized where-used list data for the item Root. The default value is _BOMOUT_. A null value is replaced by the default value.

DROP=*variables*
specifies all variables in the Indented BOM data set that you do not want to be included in the output data set. The default value is the list that contains the variables specified in the LEV=, QUANTITY=, QTYPROD=, and RID= parameters, in addition to the _Parent_, Paren_ID, Part_ID, _Prod_, Tot_Lead, and Tot_Off variables, if they are in the Indented BOM data set. A null value tells the macro program to write all the variables in the Indented BOM data set to the output data set.

The Input Data Set

The input data set of this macro is the Indented BOM data set described in "Input Data Sets" on page 82. The following parameters specify the variables in the input data set that are required by the %BOMRSUW macro.

COMP=	A character or numeric variable that contains the part number of the component.
LEV=	A numeric variable that contains the indenture level information.
QTYPROD=	A numeric variable that contains the quantity per product information.

The following parameters specify the variables in the Indented BOM input data set that are used by the macro, if they can be found in the data set.

QUANTITY= A numeric variable that contains the quantity per assembly information. This variable is only needed in creating the default drop variables.

RID= Variables that contain product structure information, except for those specified in the COMP= or QUANTITY= parameters. These variables are only needed in creating the default drop variables.

DROP= Variables that you do not want to be included in the output data set. These variables are optional.

The Output Data Set

The output data set contains the summarized where-used list for the item Root. This data set has the same variables as the Indented BOM data set, except for the variables specified in the DROP= parameter. The output data set contains a new variable with the name specified by the following parameter:

QTYUSED= A numeric variable that contains the total quantities of the item Root used, directly or indirectly, in the item identified by the Component variable.

Macro Variable _BOMRSUW_

Upon completion, this macro defines a macro variable, _BOMRSUW_, which contains a character string that indicates the status of the macro execution. See "Macro Variables" on page 85 for the possible values of this macro variable.

An Example

The following statement constructs a summarized where-used list for the item '1400' from the Summarized BOM data set as displayed in Figure 4.1.

```
%bomrsuw(root='1400', in=IndBOM, quantity=QtyPer,
qtyused=QtyUsed, rid=SDate EDate LTOff Fscrap,
out=BomOut6);
```

The summarized where-used list for the item '1400' is displayed in Figure 4.7. Note that a summarized where-used lists a parent item only once for the total quantity of the given component (the item '1400' in this example) used by the parent item. For example, the item 'LA01' uses 6 units of the item '1400', 4 units for making the item 'B100' and 2 for making '1500'. However, the summarized where-used list displayed in Figure 4.7 shows that the item '1500' uses 4 units of '1400'. This is because another 2 units are used in the bill of material for the end item 'A101'.

Figure 4.7 Summarized Where-Used List for Item 1400 (BomOut6)

ABC Lamp Company
%BOMRSUW macro
Summarized Where-Used List, Part 1400

Part	Desc	Unit	Lead Time	Qty Used
1500	Steel holder	Each	2	4
A100	One-way socket assem.	Each	1	2
A101	Three-way socket assem.	Each	1	2
B100	Base assembly	Each	1	4
LA01	Lamp LA	Each	2	6

Transactional Macros

The SAS macros described in this section perform specialized transactions for reducing the clerical time required to maintain the bill of material. These macros update Indented BOM data sets and are not concerned with the original Part Master and Product Structure data sets. Some companies use these two data sets to create an Indented BOM data set. Thereafter, they use, maintain, or update only the Indented BOM data set without referring again to the Part Master and Product Structure data sets.

The transactional macros enable you to make changes to the indented bill of material, such as replacing an item with another one, deleting an item, and so on. These macros should be used carefully. Used correctly, they enable you to avoid some clerical work and can be helpful in certain situations. However, if the macros are used incorrectly, it may take hours to reconstruct what was changed automatically in a matter of seconds (Landvater and Gray 1989).

The macros contained in this section are as follows:

%BOMTCNP performs copy-and-paste transaction

%BOMTDEL performs multiple-delete transaction

%BOMTREP performs multiple-replace transaction

%BOMTSAE performs same-as-except transaction

The *copy-and-paste* transaction copies a bill of material for a given item and attaches it to another parent item. This is normally followed by some transactions to change a few of the components in the copied bill of material. The *multiple-delete* transaction removes the entire bill of material for a given item in all its appearances. This transaction is used most often when an item is obsolete. The *multiple-replace* transaction searches the entire data set and replaces the bill of material for a given item with another in all its uses. This transaction is used when one item is replacing another in every bill of material where the original item is used (Landvater and Gray 1989). The *same-as-except* transaction creates a new bill of material that is a clone of an old one, and then replaces one item with another in all its appearances in the newly created bill.

%BOMTCNP: Copy-and-Paste Transaction

%bomtcnp (*ROOT=root, ATTACH=attach, IN=SAS-data-set,*
PSDATA=SAS-data-set, PARENT=variable, PARENID=variable,
COMP=variable, QUANTITY=variable, OFFSET=variable,
FACTOR=variable, LEADTIME=variable, ID=variables,
RID=variables, OUT=SAS-data-set);

performs the copy-and-paste transaction, which copies the bill of material for the item specified in the ROOT= parameter (referred to as the item Root), and attaches it to the parent item specified in the ATTACH= parameter (referred to as the item Attach). The parameters for this macro are as follows:

ROOT=*root*

specifies the part number of the item for which the bill of material is copied. This item is referred to as the item Root. You must specify either a number or a quoted string for the ROOT= parameter, depending on the type of the Component variable.

ATTACH=*attach*

specifies the part number of the parent item to which the copied bill of material is attached. This item is referred to as the item Attach. You must specify either a number or a quoted string for the ATTACH= parameter, depending on the type of the Component variable.

IN=*SAS-data-set*

names the Indented BOM input data set. The default value is the most recently created SAS data set. A null value is replaced by the default value.

PSDATA=*SAS-data-set*

names the Product Structure input data set that contains the product structure record for the relationship between the parent item Attach and the component Root. The default value is _BOMPS_. A null value is replaced by the default value. If the data set specified in the PSDATA= parameter does not exist, the macro will create a data set _BOMPS_ which contains the product structure record for the relationship between the item Attach and the item Root only. In this product structure record, the quantity per assembly is 1, and both the lead-time offset and the scrap factor are 0. The values of all RID variables for the product structure record are missing.

PARENT=*variable*

identifies the variable in the Indented BOM and the Product Structure data sets that contains the parent item's part number. The default value is _Parent_. A null value is replaced by the default value.

PARENID=*variable*

identifies the variable in the Indented BOM data set that contains the ID number for the parent record. The default value is Paren_ID. A null value is replaced by the default value.

COMP=*variable*

identifies the variable in the Indented BOM and the Product Structure data sets that contains the component's part number. This variable is referred to as the Component variable. The default value is _Part_. A null value is replaced by the default value.

QUANTITY=*variable*
identifies the variable in the Indented BOM and the Product Structure data sets that contains the quantity per assembly information. A null value tells the macro program not to look for the variable in both the Indented BOM and the Product Structure data sets that contains the quantity per assembly information. The default value is Qty_Per.

OFFSET=*variable*
identifies the variable in the Indented BOM and the Product Structure data sets that contains the lead-time offset information. A null value tells the macro program not to look for the variable in both the Indented BOM and the Product Structure data sets that contains the lead-time offset information. The default value is L_Offset.

FACTOR=*variable*
identifies the variable in the Indented BOM and the Product Structure data sets that contains the scrap factor information. A null value tells the macro program not to look for the variable in both the Indented BOM and the Product Structure data sets that contains the scrap factor information. The default value is S_Factor.

LEADTIME=*variable*
identifies the variable in the Indented BOM data set that contains the lead time information. If you do not specify this parameter or specify a null value, the macro program will not look for the variable in the input data set that contains the lead time information.

ID=*variables*
identifies all ID variables in the Indented BOM data set that contain part master data and are not specified in the COMP= or LEADTIME= parameters. See "Part Master Data Set" on page 29 and "STRUCTURE Statement" on page 25, both in Chapter 3 for details about the part master data and the ID variables. If you do not specify this parameter or specify a null value, the macro program assumes the Indented BOM data set contains no ID variables.

RID=*variables*
identifies all RID variables in the Indented BOM and Product Structure data sets that contain product structure data, and are not specified in the PARENT=, COMP=, QUANTITY=, FACTOR=, or OFFSET= parameters. See "Product Structure Data Set" on page 30 and "STRUCTURE Statement" on page 25, both in Chapter 3 for details about the product structure data and the RID variables. If you do not specify this parameter or specify a null value, the macro program assumes both the Indented BOM and the Product Structure data sets contain no RID variables.

OUT=*SAS-data-set*
names the output data set that contains the modified indented bill of material. The default value is _BOMOUT_. A null value is replaced by the default value.

The Input Data Sets

The input data sets of this macro are the Indented BOM and the Product Structure data sets described in "Input Data Sets" on page 82. The Indented BOM data set contains the bill of material for all end items of the product line, plant, or company.

The following parameters specify the variables in the Indented BOM input data set that are required by the %BOMTCNP macro.

COMP= A character or numeric variable that contains the part number of the component.

PARENID= A numeric variable that contains the ID number for the parent record.

PARENT= A character or numeric variable that contains the part number of the parent item. The type of this variable must be the same as that of the variable specified in the COMP= parameter.

The following parameters specify the variables in the Indented BOM input data set that are used by the macro, if they can be found in the data set.

FACTOR= A numeric variable that contains the scrap factor information.

ID= Variables that contain part master information, except for those specified in the COMP= or LEADTIME= parameters.

LEADTIME= A numeric variable that contains the lead time of the component.

OFFSET= A numeric variable that contains the lead-time offset information.

QUANTITY= A numeric variable that contains the quantity per assembly information.

RID= Variables that contain product structure information, except for those specified in the COMP=, FACTOR=, OFFSET=, PARENT=, or QUANTITY= parameters.

The copy-and-paste transaction creates a new parent-component relationship with the item Attach as the parent item and the item Root as the component. The product structure data for this new relationship usually cannot be found in the Indented BOM input data set. Therefore, this macro needs a Product Structure input data set that contains the product structure record for the new relationship.

The following parameters specify the variables in the Product Structure data set that are required by the %BOMTCNP macro.

COMP= A character or numeric variable that contains the part number of the component.

PARENT= A character or numeric variable that contains the part number of the parent item. The type of this variable must be the same as that of the variable specified in the COMP= parameter.

The following parameters specify the variables in the Product Structure input data set that are used by the macro, if they can be found in the data set.

FACTOR= A numeric variable that contains the scrap factor information. If the macro cannot find this variable in the Product Structure data set, it assumes the scrap factor of the new relationship is 0.

OFFSET= A numeric variable that contains the lead-time offset information. If the macro cannot find this variable in the Product Structure data set, it assumes the lead-time offset of the new relationship is 0.

QUANTITY= A numeric variable that contains the quantity per assembly information. If the macro cannot find this variable in the Product Structure data set, it assumes the quantity per assembly of the new relationship is 1.

RID= Variables that contain product structure information, except for those specified in the COMP=, FACTOR=, OFFSET=, PARENT=, or QUANTITY= parameters. If the macro cannot find an RID variable in the Product Structure data set, but it is in the Indented BOM data set, it assumes the value of this variable is missing.

The Output Data Set

The output data set of this macro contains all the bill of material in the input data set, with the bill of material for the item Root that is copied and attached to the parent item Attach. This transaction is done by extracting the Product Structure data from the Indented BOM input data set. Thereafter, the product structure record for the new relationship with the item Attach as the parent item and the item Root as the component is added to this data set. The product structure record for this new relationship should be contained in the input data set with the name specified by the PSDATA= parameter. The BOM procedure is then invoked with the new Product Structure data set to create the new indented bill of material. Therefore, the output data set of this macro has the same variables as the Indented BOM output data set of the BOM procedure. See "Indented BOM Data Set" on page 32 in Chapter 3 for the variables in this data set.

Macro Variable _BOMTCNP_

Upon completion, this macro defines a macro variable, _BOMTCNP_, which contains a character string that indicates the status of the macro execution. See "Macro Variables" on page 85 for the possible values of this macro variable.

An Example

In this example, we assume the engineers of the ABC Lamp Company decide that the wiring assembly (part number '1700') also needs one 1/4-20 screw (part number '1400'). To modify the company's bill of material data, they need to create a new product structure record for the relationship of the parent item '1700' and the component '1400'. The Product Structure input data set, ParComp1, displayed in Figure 4.8, provides the product structure record for this new relationship.

```
proc print data=ParComp1 noobs;
var _Parent_ _Part_ QtyPer Fscrap LTOff SDate EDate;
title 'ABC Lamp Company';
title3 'Product Structure Record; Parent 1700, Component 1400';
run;
```

Figure 4.8 Product Structure Record for the Relationship 1700-1400 (ParComp1)

ABC Lamp Company

Product Structure Record; Parent 1700, Component 1400

Parent	_Part_	Qty Per	Fscrap	LTOff	SDate	EDate
1700	1400	1	.	.	.	.

The following SAS code invokes the %BOMTCNP macro to perform the transaction.

```
%bomtcnp(root='1400', attach='1700', in=IndBOM, psdata=ParComp1,
quantity=QtyPer, offset=LTOff, factor=Fscrap,
leadtime=LeadTime, id=Desc Unit, rid=SDate EDate,
out=BomOut7);
```

The data set BomOut7 displayed in Figure 4.9 contains the new indented bill of material. Note that the item '1400' is now used by the item '1700' in the bill of material for both end items 'A101' and 'LA01'.

Figure 4.9 Indented Bill of Material Updated by the Copy-And-Paste Transaction (BomOut7)

ABC Lamp Company
%BOMTCNP macro
Indented Bill of Material

Obs	_Level_	_Parent_	Paren_ID	_Part_	Part_ID	Desc	Qty Per	Fscrap
1	0		.	A101	0	Three-way socket assem.	.	.
2	1	A101	0	1500	1	Steel holder	1	0.0
3	2	1500	1	1400	2	1/4-20 Screw	2	0.0
4	1	A101	0	1601	3	Three-way socket	1	0.0
5	1	A101	0	1700	4	Wiring assembly	1	0.0
6	2	1700	4	2200	5	16-Gauge lamp cord	12	0.1
7	2	1700	4	2210	6	14-Gauge lamp cord	12	0.1
8	2	1700	4	2300	7	Standard plug terminal	1	0.0
9	2	1700	4	1400	8	1/4-20 Screw	1	0.0
10	0		.	LA01	9	Lamp LA	.	.
11	1	LA01	9	A100	10	One-way socket assem.	1	0.0
12	2	A100	10	1500	11	Steel holder	1	0.0
13	3	1500	11	1400	12	1/4-20 Screw	2	0.0
14	2	A100	10	1600	13	One-way socket	1	0.0
15	2	A100	10	1700	14	Wiring assembly	1	0.0
16	3	1700	14	2200	15	16-Gauge lamp cord	12	0.1
17	3	1700	14	2210	16	14-Gauge lamp cord	12	0.1
18	3	1700	14	2300	17	Standard plug terminal	1	0.0
19	3	1700	14	1400	18	1/4-20 Screw	1	0.0
20	1	LA01	9	B100	19	Base assembly	1	0.0
21	2	B100	19	1100	20	Finished shaft	1	0.0
22	3	1100	20	2100	21	3/8 Steel tubing	26	0.2

Obs	Qty_Prod	Unit	Lead Time	Tot_Lead	LTOff	Tot_Off	SDate	EDate	_Prod_
1	1	Each	1	1	.	0	.	.	A101
2	1	Each	2	3	0	0	.	.	A101
3	2	Each	1	4	0	0	.	.	A101
4	1	Each	2	3	0	0	.	.	A101
5	1	Each	1	2	0	0	.	.	A101
6	12	Feet	2	4	0	0	.	07APR2001	A101
7	12	Feet	2	4	0	0	08APR2001	.	A101
8	1	Each	1	3	0	0	.	.	A101
9	1	Each	1	3	0	0	.	.	A101
10	1	Each	2	2	.	0	.	.	LA01
11	1	Each	1	3	2	2	.	.	LA01
12	1	Each	2	5	0	2	.	.	LA01
13	2	Each	1	6	0	2	.	.	LA01
14	1	Each	2	5	0	2	.	.	LA01
15	1	Each	1	4	0	2	.	.	LA01
16	12	Feet	2	6	0	2	.	07APR2001	LA01
17	12	Feet	2	6	0	2	08APR2001	.	LA01
18	1	Each	1	5	0	2	.	.	LA01
19	1	Each	1	5	0	2	.	.	LA01
20	1	Each	1	3	0	0	.	.	LA01
21	1	Each	2	5	0	0	.	.	LA01
22	26	Inches	3	8	0	0	.	.	LA01

Figure 4.9 *continued*

```
                              ABC Lamp Company
                                %BOMTCNP macro
                            Indented Bill of Material

                                                                                  Qty
Obs _Level_ _Parent_ Paren_ID _Part_ Part_ID Desc                                 Per Fscrap

 23    2      B100      19     1200     22   6-Diameter steel plate              1   0.0
 24    2      B100      19     1300     23   Hub                                 1   0.0
 25    2      B100      19     1400     24   1/4-20 Screw                        4   0.0
 26    1      LA01       9     S100     25   Black shade                         1   0.0

                      Lead
Obs Qty_Prod  Unit    Time Tot_Lead LTOff Tot_Off     SDate     EDate _Prod_

 23     1     Each     3      6      0      0            .         .  LA01
 24     1     Each     2      5      1      1            .         .  LA01
 25     4     Each     1      4      3      3            .         .  LA01
 26     1     Each     2      4      0      0            .         .  LA01
```

%BOMTDEL: Multiple-Delete Transaction

%bomtdel (*ROOT=root, IN=SAS-data-set, LEV=variable, COMP=variable, OUT=SAS-data-set*);

performs the multiple-delete transaction, which removes the entire bill of material for the item specified in the ROOT= parameter (referred to as item Root) in all its appearances. The parameters for this macro are as follows:

ROOT=*root*

specifies the part number of the item for which the bill of material is deleted from the Indented BOM data set. This item is referred to as the item Root. You must specify either a number or a quoted string for the ROOT= parameter, depending on the type of the Component variable.

IN=*SAS-data-set*

names the Indented BOM input data set. The default value is the most recently created SAS data set. A null value is replaced by the default value.

LEV=*variable*

identifies the variable in the Indented BOM data set that contains the indenture level information. The default value is _Level_. A null value is replaced by the default value.

COMP=*variable*

identifies the variable in the Indented BOM data set that contains the component's part number. This variable is referred to as the Component variable. The default value is _Part_. A null value is replaced by the default value.

OUT=*SAS-data-set*
names the output data set that contains the modified indented bill of material. The default value is _BOMOUT_. A null value is replaced by the default value.

The Input Data Set

The input data set of this macro is the Indented BOM data set described in "Input Data Sets" on page 82.

The following parameters specify the variables in the input data set that are required by the %BOMTDEL macro.

COMP= A character or numeric variable that contains the part number of the component.

LEV= A numeric variable that contains the indenture level information.

The Output Data Set

The output data set of this macro contains all the bill of material in the input data set except for the complete bill of material for the item Root, which is removed from the input data set in all places where the item Root appears. This data set has the same variables as the Indented BOM input data set.

Macro Variable _BOMTDEL_

Upon completion, this macro defines a macro variable, _BOMTDEL_, which contains a character string that indicates the status of the macro execution. See "Macro Variables" on page 85 for the possible values of this macro variable.

An Example

In this example, the engineers of the ABC Lamp Company decide to retire the item '2200'. The following code invokes the %BOMTDEL macro to remove the item '2200' from the Indented BOM data set, IndBOM, as displayed in Figure 4.1.

```
%bomtdel(root='2200', in=IndBOM, out=BomOut8);
```

The data set BomOut8 displayed in Figure 4.10 contains the updated indented bill of material of the company. Note that the item '2200' is removed from both bill of material for the end items '′A101 and 'LA01'.

Figure 4.10 Indented Bill of Material Updated by the Multiple-Delete Transaction (BomOut8)

ABC Lamp Company
%BOMTDEL macro
Indented Bill of Material

Obs	_Level_	_Parent_	Paren_ID	_Part_	Part_ID	Desc	Qty Per	Fscrap
1	0		.	A101	0	Three-way socket assem.	.	.
2	1	A101	0	1500	1	Steel holder	1	0.0
3	2	1500	1	1400	2	1/4-20 Screw	2	0.0
4	1	A101	0	1601	3	Three-way socket	1	0.0
5	1	A101	0	1700	4	Wiring assembly	1	0.0
6	2	1700	4	2210	6	14-Gauge lamp cord	12	0.1
7	2	1700	4	2300	7	Standard plug terminal	1	0.0
8	0		.	LA01	8	Lamp LA	.	.
9	1	LA01	8	A100	9	One-way socket assem.	1	0.0
10	2	A100	9	1500	10	Steel holder	1	0.0
11	3	1500	10	1400	11	1/4-20 Screw	2	0.0
12	2	A100	9	1600	12	One-way socket	1	0.0
13	2	A100	9	1700	13	Wiring assembly	1	0.0
14	3	1700	13	2210	15	14-Gauge lamp cord	12	0.1
15	3	1700	13	2300	16	Standard plug terminal	1	0.0
16	1	LA01	8	B100	17	Base assembly	1	0.0
17	2	B100	17	1100	18	Finished shaft	1	0.0
18	3	1100	18	2100	19	3/8 Steel tubing	26	0.2
19	2	B100	17	1200	20	6-Diameter steel plate	1	0.0
20	2	B100	17	1300	21	Hub	1	0.0
21	2	B100	17	1400	22	1/4-20 Screw	4	0.0
22	1	LA01	8	S100	23	Black shade	1	0.0

Obs	Qty_Prod	Unit	Lead Time	Tot_Lead	LTOff	Tot_Off	SDate	EDate	_Prod_
1	1	Each	1	1	.	0	.	.	A101
2	1	Each	2	3	0	0	.	.	A101
3	2	Each	1	4	0	0	.	.	A101
4	1	Each	2	3	0	0	.	.	A101
5	1	Each	1	2	0	0	.	.	A101
6	12	Feet	2	4	0	0	08APR2001	.	A101
7	1	Each	1	3	0	0	.	.	A101
8	1	Each	2	2	.	0	.	.	LA01
9	1	Each	1	3	2	2	.	.	LA01
10	1	Each	2	5	0	2	.	.	LA01
11	2	Each	1	6	0	2	.	.	LA01
12	1	Each	2	5	0	2	.	.	LA01
13	1	Each	1	4	0	2	.	.	LA01
14	12	Feet	2	6	0	2	08APR2001	.	LA01
15	1	Each	1	5	0	2	.	.	LA01
16	1	Each	1	3	0	0	.	.	LA01
17	1	Each	2	5	0	0	.	.	LA01
18	26	Inches	3	8	0	0	.	.	LA01
19	1	Each	3	6	0	0	.	.	LA01
20	1	Each	2	5	1	1	.	.	LA01
21	4	Each	1	4	3	3	.	.	LA01
22	1	Each	2	4	0	0	.	.	LA01

%BOMTREP: Multiple-Replace Transaction

%bomtrep (*ROOT=root, REPBY=repby, IN=SAS-data-set,*
LEV=variable, PARENT=variable, PARENID=variable,
COMP=variable, QUANTITY=variable, OFFSET=variable,
FACTOR=variable, LEADTIME=variable, ID=variables,
RID=variables, DEL=del, OUT=SAS-data-set);

performs the multiple-replace transaction, which replaces the bill of material for the item specified in the ROOT= parameter (referred to as the item Root) with the bill of material for the item specified in the REPBY= parameter (referred to as the item Repby) in all places where the item Root is used. The parameters for this macro are as follows:

ROOT=*root*

specifies the part number of the item for which the bill of material is replaced with the one for the item Repby in all places where it is used. This item is referred to as the item Root. You must specify either a number or a quoted string for the ROOT= parameter, depending on the type of the Component variable.

REPBY=*repby*

specifies the part number of the item for which the bill of material is replacing the one for the item Root in all places it is used. This item is referred to as the item Repby. You must specify either a number or a quoted string for the REPBY= parameter, depending on the type of the Component variable.

IN=*SAS-data-set*

names the Indented BOM input data set. The default value is the most recently created SAS data set. A null value is replaced by the default value.

LEV=*variable*

identifies the variable in the Indented BOM data set that contains the indenture level information. The default value is _Level_. A null value is replaced by the default value.

PARENT=*variable*

identifies the variable in the Indented BOM data set that contains the parent item's part number. The default value is _Parent_. A null value is replaced by the default value.

PARENID=*variable*

identifies the variable in the Indented BOM data set that contains the ID number for the parent record. The default value is Paren_ID. A null value is replaced by the default value.

COMP=*variable*

identifies the variable in the Indented BOM data set that contains the component's part number. The default value is _Part_. A null value is replaced by the default value.

QUANTITY=*variable*

identifies the variable in the Indented BOM data set that contains the quantity per assembly information. A null value tells the macro program not to look for the variable that contains the quantity per assembly information. The default value is Qty_Per.

OFFSET=*variable*
identifies the variable in the Indented BOM data set that contains the lead-time offset information. A null value tells the macro program not to look for the variable that contains lead-time offset information. The default value is L_Offset.

FACTOR=*variable*
identifies the variable in the Indented BOM data set that contains the scrap factor information. A null value tells the macro program not to look for the variable that contains the scrap factor information. The default value is S_Factor.

LEADTIME=*variable*
identifies the variable in the Indented BOM data set that contains the lead time information. If you do not specify this parameter or specify a null value, the macro program will not look for the variable in the input data set that contains the lead time information.

ID=*variables*
identifies all ID variables in the Indented BOM data set that contain part master data and are not specified in the COMP= or LEADTIME= parameters. See "Part Master Data Set" on page 29 and "STRUCTURE Statement" on page 25, both in Chapter 3 for details about the part master data and the ID variables. If you do not specify this parameter or specify a null value, the macro program assumes the Indented BOM data set contains no ID variables.

RID=*variables*
identifies all RID variables in the Indented BOM data set that contain product structure data and are not specified in the PARENT=, COMP=, QUANTITY=, FACTOR=, or OFFSET= parameters. See "Product Structure Data Set" on page 30 and "STRUCTURE Statement" on page 25, both in Chapter 3 for details about the product structure data and the RID variables. If you do not specify this parameter or specify a null value, the macro program assumes the Indented BOM data set contains no RID variables.

DEL=*del*
specifies either 0 or 1 to indicate whether or not the bill of material for the item Root should be removed after the replace transaction is done. The value 0 indicates that the bill of material should not be removed; any other value indicates that the bill of material should be removed. The default value is 0. A null value is replaced with the default value.

OUT=*SAS-data-set*
names the output data set that contains the modified indented bill of material. The default value is _BOMOUT_. A null value is replaced by this default value.

The Input Data Set

The input data set of this macro is the Indented BOM data set as described in "Input Data Sets" on page 82.

The following parameters specify the variables in the input data set that are required by the %BOMTREP macro.

COMP=	A character or numeric variable that contains the part number of the component.
LEV=	A numeric variable that contains the indenture level information.
PARENID=	A numeric variable that contains the ID number for the parent record.

PARENT= A character or numeric variable that contains the part number of the parent item. The type of this variable must be the same as that of the variable specified in the COMP= parameter.

The following parameters specify the variables in the Indented BOM input data set that are used by the macro, if they can be found in the data set.

FACTOR= A numeric variable that contains the scrap factor information.

ID= Variables that contain part master information, except for those specified in the COMP= or LEADTIME= parameters.

LEADTIME= A numeric variable that contains the lead time of the component.

OFFSET= A numeric variable that contains the lead-time offset information.

QUANTITY= A numeric variable that contains the quantity per assembly information.

RID= Variables that contain product structure information, except for those specified in the COMP=, FACTOR=, OFFSET=, PARENT=, or QUANTITY= parameters.

The Output Data Set

The output data set contains all the bill of material in the input data set with the bill of material for the item Root replaced by the bill of material for the item Repby. This output data set has the same variables as the Indented BOM output data set of the BOM procedure. See "Indented BOM Data Set" on page 32 in Chapter 3 for the variables in this data set.

Macro Variable _BOMTREP_

Upon completion, this macro defines a macro variable, _BOMTREP_, which contains a character string that indicates the status of the macro execution. See "Macro Variables" on page 85 for the possible values of this macro variable.

An Example

In this example, the engineers of the ABC Lamp Company want to replace the one-way socket assembly (part number '`A100`') with the three-way socket assembly (part number '`A101`') in all places where '`A100`' is used. The following code performs this multiple-replace transaction.

```
%bomtrep(root='A100', repby='A101', in=IndBOM, quantity=QtyPer,
offset=LTOff, factor=Fscrap, leadtime=LeadTime,
id=Desc Unit, rid=SDate EDate, del=1, out=BomOut9);
```

The data set BomOut9 displayed in Figure 4.11 contains the updated indented bill of material of the company. Note that the item '`A100`' is replaced by the item '`A101`' in the bill of material for the end item '`LA01`'. The bill of material for the item '`A100`' is removed from the company's product structure because the parameter DEL=1 is specified in the invocation of the %BOMTREP macro.

Figure 4.11 Indented Bill of Material Updated by the Multiple-Replace Transaction (BomOut9)

ABC Lamp Company
%BOMTREP macro
Indented Bill of Material

Obs	_Level_	_Parent_	Paren_ID	_Part_	Part_ID	Desc	Qty Per	Fscrap
1	0		.	LA01	8	Lamp LA	.	.
2	1	LA01	8	A101	9	Three-way socket assem.	1	0.0
3	2	A101	9	1500	10	Steel holder	1	0.0
4	3	1500	10	1400	11	1/4-20 Screw	2	0.0
5	2	A101	9	1601	12	Three-way socket	1	0.0
6	2	A101	9	1700	13	Wiring assembly	1	0.0
7	3	1700	13	2200	14	16-Gauge lamp cord	12	0.1
8	3	1700	13	2210	15	14-Gauge lamp cord	12	0.1
9	3	1700	13	2300	16	Standard plug terminal	1	0.0
10	1	LA01	8	B100	17	Base assembly	1	0.0
11	2	B100	17	1100	18	Finished shaft	1	0.0
12	3	1100	18	2100	19	3/8 Steel tubing	26	0.2
13	2	B100	17	1200	20	6-Diameter steel plate	1	0.0
14	2	B100	17	1300	21	Hub	1	0.0
15	2	B100	17	1400	22	1/4-20 Screw	4	0.0
16	1	LA01	8	S100	23	Black shade	1	0.0

Obs	Qty_Prod	Unit	Lead Time	Tot_Lead	LTOff	Tot_Off	SDate	EDate	_Prod_
1	1	Each	2	2	.	0	.	.	LA01
2	1	Each	1	3	2	2	.	.	LA01
3	1	Each	2	5	0	2	.	.	LA01
4	2	Each	1	6	0	2	.	.	LA01
5	1	Each	2	5	0	2	.	.	LA01
6	1	Each	1	4	0	2	.	.	LA01
7	12	Feet	2	6	0	2	.	07APR2001	LA01
8	12	Feet	2	6	0	2	08APR2001	.	LA01
9	1	Each	1	5	0	2	.	.	LA01
10	1	Each	1	3	0	0	.	.	LA01
11	1	Each	2	5	0	0	.	.	LA01
12	26	Inches	3	8	0	0	.	.	LA01
13	1	Each	3	6	0	0	.	.	LA01
14	1	Each	2	5	1	1	.	.	LA01
15	4	Each	1	4	3	3	.	.	LA01
16	1	Each	2	4	0	0	.	.	LA01

%BOMTSAE: Same-as-Except Transaction

%bomtsae (*ROOT=root, SAMEAS=sameas, EXCEPT=except,*
REPBY=repby, IN=SAS-data-set, PMDATA=SAS-data-set,
PART=variable, LEV=variable, PARENT=variable,
PARENID=variable, COMP=variable, QUANTITY=variable,
OFFSET=variable, FACTOR=variable, LEADTIME=variable,
ID=variables, RID=variables, OUT=SAS-data-set);

performs the same-as-except transaction, which creates a new indented bill of material for the item specified in the ROOT= parameter (referred to as the item Root), which is a clone of the bill of material for the item specified in the SAMEAS= parameter (referred to as the item Sameas). Then, it replaces the item specified in the EXCEPT= parameter (referred to as the item Except) with the item specified in the REPBY= parameter (referred to as the item Repby) in the newly created bill of material. Note that, unlike other transactional macros, the output data set of this macro contains the new bill of material for the item Root only. In addition, if the item Root is in the Indented BOM input data set, the bill of material for the item Root in the input data set may be different from the one in the output data set. The parameters for this macro are as follows:

ROOT=*root*

specifies the part number of the item for which the bill of material is a clone of the bill of material for the item Sameas. This item is referred to as the item Root. You must specify either a number or a quoted string for the ROOT= parameter, depending on the type of the Component variable. If you do not specify this parameter or specify a null value, the macro will only perform the "replace" transaction, which replaces the item Except with the item Repby in the bill of material for the item Sameas.

SAMEAS=*sameas*

specifies the part number of the item for which the bill of material is cloned. This item is referred to as the item Sameas. You must specify either a number or a quoted string for the SAMEAS= parameter, depending on the type of the Component variable.

EXCEPT=*except*

specifies the part number of the item for which the bill of material is replaced by the bill of material for item Repby in all its uses. This item is referred to as the item Except. You must specify either a number or a quoted string for the EXCEPT= parameter, depending on the type of the Component variable. If you do not specify this parameter or specify a null value, the macro will only perform the "clone" transaction, which copies the bill of material for the item Sameas to the item Root. In other words, the indented bill of material in the output data set is identical to the indented bill of material for the item Sameas in the Indented BOM input data set.

REPBY=*repby*

specifies the part number of the item for which the bill of material is replacing the bill of material for item Except. This item is referred to as the item Repby. You must specify either a number or a quoted string for the REPBY= parameter, depending on the type of the Component variable. If you do not specify this parameter or specify a null value, the macro will remove the bill of material for the item Except in all its appearances in the new indented bill of material.

IN=*SAS-data-set*
names the Indented BOM input data set. The default value is the most recently created SAS data set. A null value is replaced by the default value.

PMDATA=*SAS-data-set*
names the Part Master input data set that contains the part master data for the item Root. The default value is the name specified by the IN= parameter. A null value is replaced by the default value. The PMDATA= parameter is ignored if you do not specify the ROOT= parameter, or specify a null value for it.

PART=*variable*
identifies the variable in the Part Master data set that contains the item's part number. The default is the variable specified in the COMP= parameter. A null value is replaced by the default value.

LEV=*variable*
identifies the variable in the Indented BOM data set that contains the indenture level information. The default value is _Level_. A null value is replaced by the default value.

PARENT=*variable*
identifies the variable in the Indented BOM data set that contains the parent item's part number. The default value is _Parent_. A null value is replaced by the default value.

PARENID=*variable*
identifies the variable in the Indented BOM data set that contains the ID number for the parent record. The default value is Paren_ID. A null value is replaced by the default value.

COMP=*variable*
identifies the variable in the Indented BOM data set that contains the component's part number. This variable is referred to as the Component variable. The default value is _Part_. A null value is replaced by the default value.

QUANTITY=*variable*
identifies the variable in the Indented BOM data set that contains the quantity per assembly information. A null value tells the macro program not to look for the variable that contains the quantity per assembly information. The default value is Qty_Per.

OFFSET=*variable*
identifies the variable in the Indented BOM data set that contains the lead-time offset information. A null value tells the macro program not to look for the variable that contains lead-time offset information. The default value is L_Offset.

FACTOR=*variable*
identifies the variable in the Indented BOM data set that contains the scrap factor information. A null value tells the macro program not to look for the variable that contains the scrap factor information. The default value is S_Factor.

LEADTIME=*variable*
identifies the variable in the Indented BOM and Part Master data sets that contains the lead time information. If you do not specify this parameter or specify a null value, the macro program will not look for the variable in the Indented BOM data set that contains the lead time information.

ID=*variables*

identifies all ID variables in the Indented BOM and Part Master data sets that contain part master data and are not specified in the PART=, COMP=, or LEADTIME= parameters. See "Part Master Data Set" on page 29 and "STRUCTURE Statement" on page 25, both in Chapter 3 for details about the part master data and the ID variables. If you do not specify this parameter or specify a null value, the macro program assumes the Indented BOM data set contains no ID variables.

RID=*variables*

identifies all RID variables in the Indented BOM data set that contain product structure data and are not specified in the PARENT=, COMP=, QUANTITY=, FACTOR=, or OFFSET= parameters. See "Product Structure Data Set" on page 30 and "STRUCTURE Statement" on page 25, both in Chapter 3 for details about the product structure data and the RID variables. If you do not specify this parameter or specify a null value, the macro program assumes the Indented BOM data set contains no RID variables.

OUT=*SAS-data-set*

names the output data set that contains the new indented bill of material for the item Root. The default value is _BOMOUT_. A null value is replaced by the default value.

Input Data Sets

The input data sets of this macro are the Indented BOM and the Part Master data sets described in "Input Data Sets" on page 82. The Indented BOM data set contains the bill of material for all end items of the product line, plant, or company.

The following parameters specify the variables the Indented BOM data set that are required by the %BOMTSAE macro.

COMP=	A character or numeric variable that contains the part number of the component.
LEV=	A numeric variable that contains the indenture level information.
PARENID=	A numeric variable that contains the ID number for the parent record.
PARENT=	A character or numeric variable that contains the part number of the parent item. The type of this variable must be the same as that of the variable specified in the COMP= parameter.

The following parameters specify the variables in the Indented BOM input data set that are used by the macro, if they can be found in the data set.

FACTOR=	A numeric variable that contains the scrap factor information.
ID=	Variables that contain part master information, except for those specified in the COMP= or LEADTIME= parameters.
LEADTIME=	A numeric variable that contains the lead time of the component.
OFFSET=	A numeric variable that contains the lead-time offset information.
QUANTITY=	A numeric variable that contains the quantity per assembly information.
RID=	Variables that contain product structure information, except for those specified in the COMP=, FACTOR=, OFFSET=, PARENT=, or QUANTITY= parameters.

Unlike other transactional macros, the item Root is not necessarily defined in the Indented BOM input data set. If it is not, you must specify a Part Master data set in the PMDATA= parameter that contains the part master record for the item Root. Otherwise, you can use the Indented BOM data set as the Part Master data set. The Part Master input data set may contain one part master record for the item Root only, or contain the part master records for all items, including the item Root, that are used in the company's product structure.

The following parameter specifies the variable in the Part Master data set that is required by the %BOMTSAE macro.

PART= A character or numeric variable that contains the part number of the item. The type of this variable must be the same as that of the variable specified in the COMP= parameter.

The following parameters specify the variables in the Product Structure input data set that are used by the macro, if they can be found in the data set.

ID= Variables that contain part master information, except for those specified in the PART= or LEADTIME= parameters.

LEADTIME= A numeric variable that contains the lead time of the component.

The Output Data Set

The output data set of this macro contains the bill of material for the item Root, which is a clone of the bill of material for the item Sameas, except the item Except, which is replaced by the item Repby in all its uses. The output data set has the same variables as the Indented BOM output data set of the BOM procedure. See "Indented BOM Data Set" on page 32 in Chapter 3 for the variables in this data set. Note that if the item Root is in the Indented BOM data set specified by the IN= parameter, the bill of material for the item Root in the input data set may be different from the one in this output data set.

Macro Variable _BOMTSAE_

Upon completion, this macro defines a macro variable, _BOMTSAE_, which contains a character string that indicates the status of the macro execution. See "Macro Variables" on page 85 for the possible values of this macro variable.

An Example

In this example, the engineers of the ABC Lamp Company need to create a new product, a lamp with three-way socket (part number 'LA03'), which is the same as the lamp 'LA01' except that the new product has a three-way socket assembly (part number 'A101'), instead of a one-way socket assembly (part number 'A100'). They first create a part master record for the new product 'LA03'. The Part Master input data set, PMaster1, displayed in Figure 4.12, contains this part master record.

```
proc print data=PMaster1 noobs;
title 'ABC Lamp Company';
title3 'Part Master Record; Part LA03';
run;
```

Figure 4.12 Part Master Record for Item LA03 (PMaster1)

ABC Lamp Company

Part Master Record; Part LA03

Part	Desc	Unit	Lead Time
LA03	Lamp LA w/ 3-way socket	Each	2

The following SAS code invokes the %BOMTSAE macro to perform the transaction.

```
%bomtsae(root='LA03', sameas='LA01', except='A100', repby='A101',
in=IndBOM, pmdata=PMaster1, part=Part, quantity=QtyPer,
offset=LTOff, factor=Fscrap, leadtime=LeadTime,
id=Desc Unit, rid=SDate EDate, out=BomOut10);
```

The data set BomOut10 that contains the indented bill of material for the new item 'LA03' is displayed in Figure 4.13.

Figure 4.13 Indented Bill of Material for Item LA03 (BomOut10)

ABC Lamp Company
%BOMTSAE macro
Indented Bill of Material, Part LA03

Obs	_Level_	_Parent_	Paren_ID	_Part_	Part_ID	Desc	Qty Per	Fscrap
1	0		.	LA03	8	Lamp LA w/ 3-way socket	.	.
2	1	LA03	8	A101	9	Three-way socket assem.	1	0.0
3	2	A101	9	1500	10	Steel holder	1	0.0
4	3	1500	10	1400	11	1/4-20 Screw	2	0.0
5	2	A101	9	1601	12	Three-way socket	1	0.0
6	2	A101	9	1700	13	Wiring assembly	1	0.0
7	3	1700	13	2200	14	16-Gauge lamp cord	12	0.1
8	3	1700	13	2210	15	14-Gauge lamp cord	12	0.1
9	3	1700	13	2300	16	Standard plug terminal	1	0.0
10	1	LA03	8	B100	17	Base assembly	1	0.0
11	2	B100	17	1100	18	Finished shaft	1	0.0
12	3	1100	18	2100	19	3/8 Steel tubing	26	0.2
13	2	B100	17	1200	20	6-Diameter steel plate	1	0.0
14	2	B100	17	1300	21	Hub	1	0.0
15	2	B100	17	1400	22	1/4-20 Screw	4	0.0
16	1	LA03	8	S100	23	Black shade	1	0.0

Obs	Qty_Prod	Unit	Lead Time	Tot_Lead	LTOff	Tot_Off	SDate	EDate	_Prod_
1	1	Each	2	2	.	0	.	.	LA03
2	1	Each	1	3	2	2	.	.	LA03
3	1	Each	2	5	0	2	.	.	LA03
4	2	Each	1	6	0	2	.	.	LA03
5	1	Each	2	5	0	2	.	.	LA03
6	1	Each	1	4	0	2	.	.	LA03
7	12	Feet	2	6	0	2	.	07APR2001	LA03
8	12	Feet	2	6	0	2	08APR2001	.	LA03
9	1	Each	1	5	0	2	.	.	LA03
10	1	Each	1	3	0	0	.	.	LA03
11	1	Each	2	5	0	0	.	.	LA03
12	26	Inches	3	8	0	0	.	.	LA03
13	1	Each	3	6	0	0	.	.	LA03
14	1	Each	2	5	1	1	.	.	LA03
15	4	Each	1	4	3	3	.	.	LA03
16	1	Each	2	4	0	0	.	.	LA03

References

Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1983), *Data Structures and Algorithms*, Reading, MA: Addison-Wesley.

Burlew, M. M. (1998), *SAS Macro Programming Made Easy*, Cary, NC: SAS Institute Inc.

Cox, J. F., III and Blackstone, J. H., Jr, eds. (1998), *APICS Dictionary*, Ninth Edition, Alexandria, VA: APICS.

Landvater, D. V. and Gray, C. D. (1989), *MRP II Standard System: A Handbook for Manufacturing Software Survival*, New York: John Wiley & Sons.

Plossl, G. (1995), *Orlicky's Material Requirements Planning*, Second Edition, New York: McGraw-Hill.

Appendix A
BOM Web Example

Contents

Overview: BOM Web Example

The BOM procedure and the macros described in this book can be used to maintain and update bills of material and produce related reports for use in the manufacturing process. The output data set produced by the BOM procedure can be used to plan for material requirements and analyze the critical path for planned orders. This appendix illustrates this process using a sample Web demo as a detailed example. In this example, 48 different items are used in the production of four final products: a 3-gallon carpet cleaner, a 4-gallon carpet cleaner, a 5-gallon carpet cleaner, and a bottled concentrate.

This application supports several different types of bills of material, providing valuable and diverse views of product makeup and part/component/material usage. It also reports on upcoming shortages and identifies the shortage status of critical parts and components. The example enables you to explore how each assembly or subassembly of a final product is manufactured, and also enables you to investigate how, where, and in what quantities any specific part or material is used. Combining this information with lead time and order information, you can identify upcoming shortages, gauge their effects on timely order fulfillment, and set procurement priorities accordingly.

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

Data: BOM Web Example

There are three main components to the data used to maintain bills of material and evaluate effects of planned orders: Part Master File, Product Structure File, and Planned Orders. This example illustrates typical forms of these data elements.

The data for the BOM Web example are directly related to the data sets required for the BOM procedure, as discussed in Chapter 3, "The BOM Procedure." The Part Master File and Planned Orders file together correspond to the PMDATA= data set, and the Product Structure File corresponds to the DATA= data set. Refer to Chapter 3 for more information about these data sets and PROC BOM.

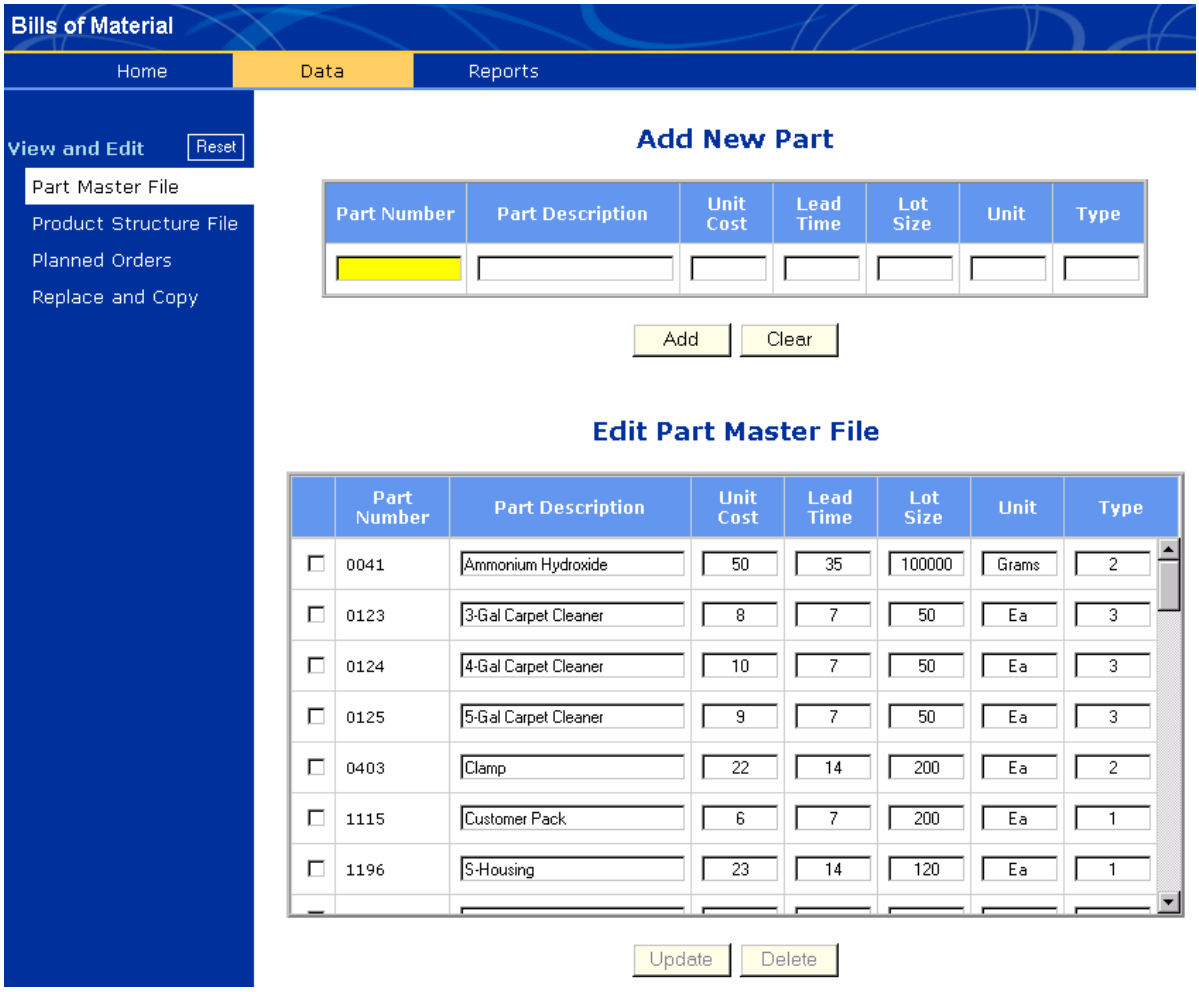
Part Master File

The Part Master File includes, for each item:

- Part number and description
- Unit cost
- Lead time
- Lot size (for replenishment)
- Unit of measure
- Item type

The Part Master File for the BOM Web example is shown in Figure A.1. Typical transactional operations required to maintain bills of material can be performed using the transactional macros described in Chapter 4, "Bill of Material Postprocessing Macros." Some of these operations are illustrated using this table. You can add a new part using the top table, update the data for existing parts using the bottom table, and delete multiple parts simultaneously. Refer to Chapter 4 for information on the BOM transactional macros.

Figure A.1 Part Master File



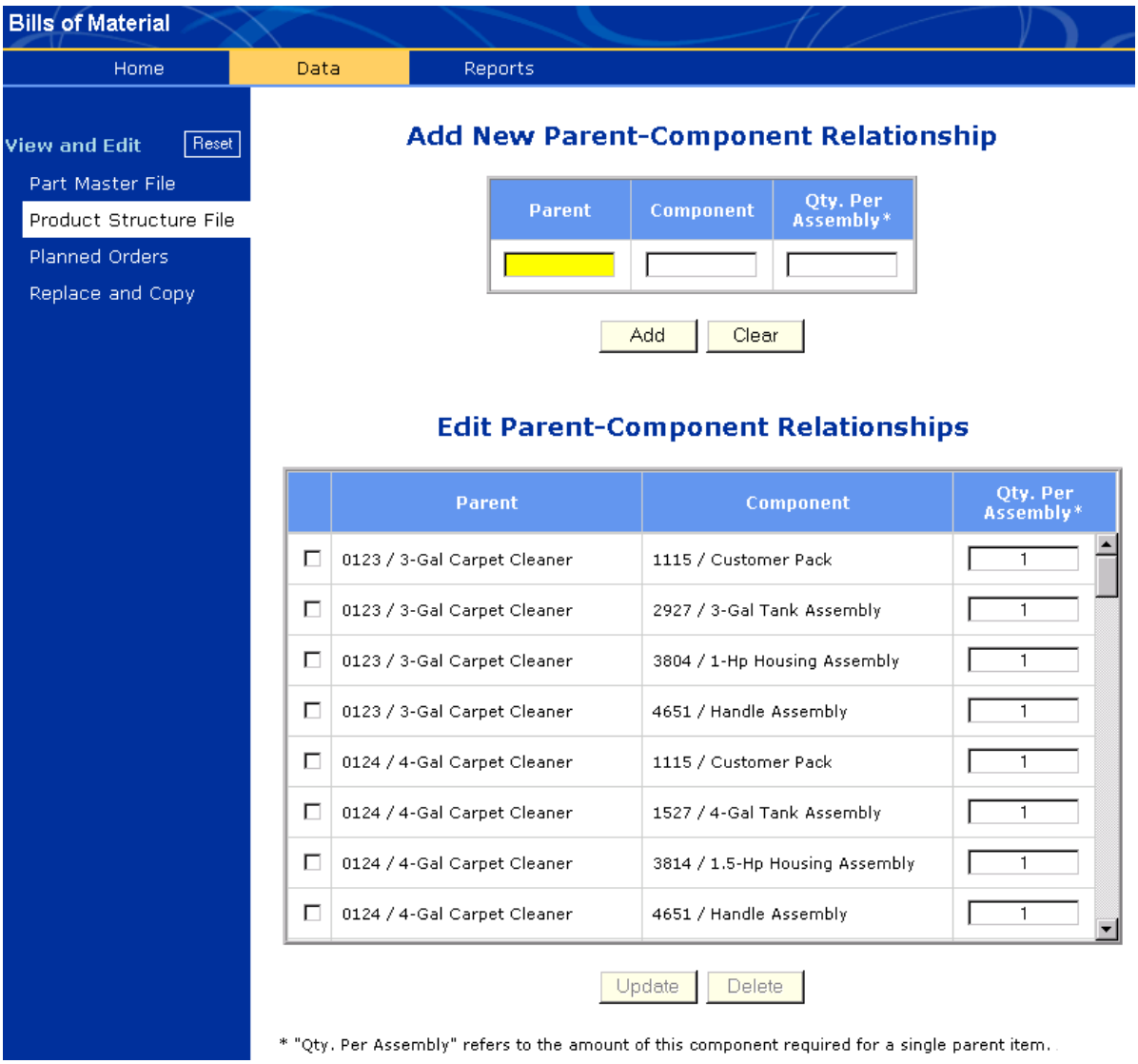
Product Structure File

The Product Structure File describes each parent-component relationship, using the following information:

- Part number for the parent item
- Part number for the component
- Quantity of the component used in the parent item

The Product Structure File for the BOM Web example is shown in Figure A.2.

Figure A.2 Product Structure File



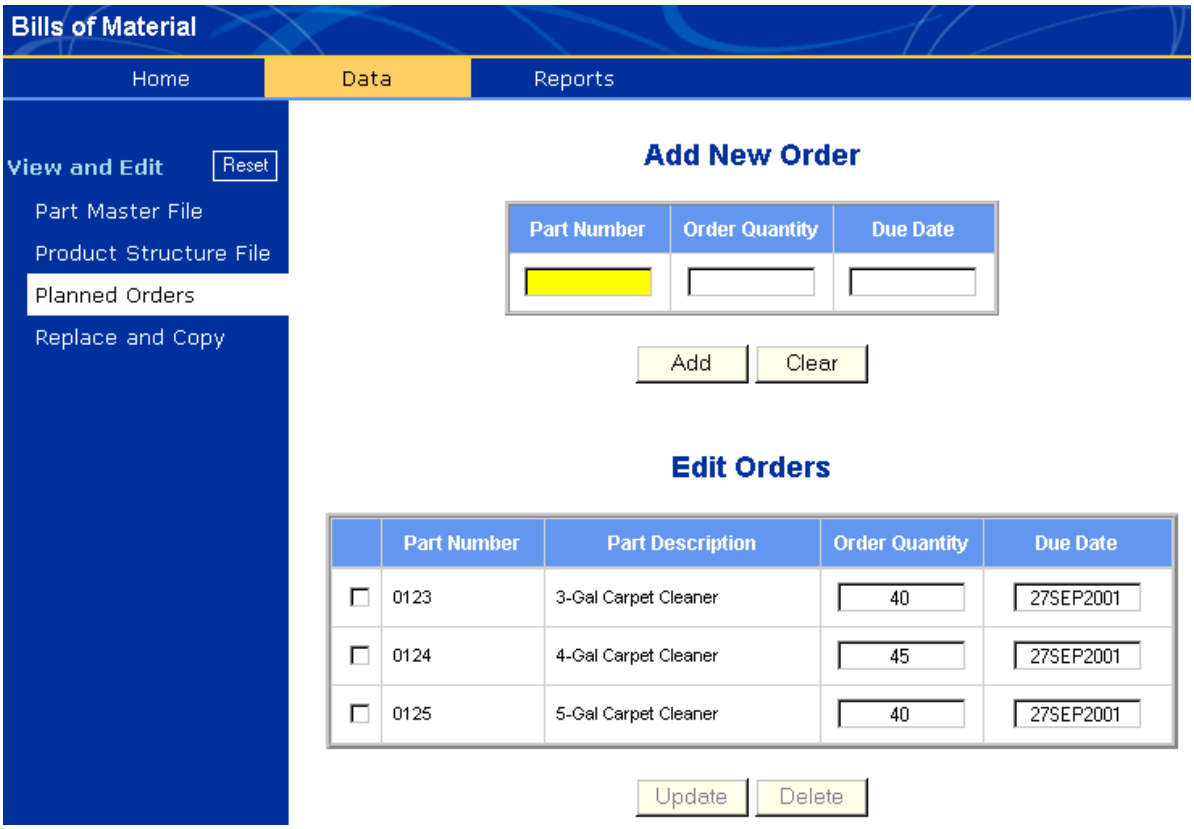
Planned Orders

To use bill of materials information in analyzing material requirements and anticipated shortages, it is necessary to keep records of all planned orders. A planned order is a future demand for a given quantity of a final product, which must be filled by a specified due date. The Planned Orders data describe each planned order, listing the following information for each order:

- Part number and description for the final product or master schedule item
- Order quantity
- Due date

The Planned Orders file for the BOM Web example is shown in Figure A.3. The planned order information is carried along to the indented bill of materials, and can be used to calculate material requirement schedules for the planned orders, as described in the section "Shortage and Critical Path Analysis Reports" on page 137.

Figure A.3 Planned Orders



Reports: BOM Web Example

Several reports can be generated using the output from the BOM procedure described in Chapter 3 and the reporting macros described in Chapter 4. This section illustrates several such reports, grouped by categories: Product Structure Reports, Miscellaneous Reports, and Shortage and Critical Path Analysis Reports

Product Structure Reports

The Product Structure Reports include a summarized parts list and indented bills of material; these reports are described in detail in the following sections.

Summarized Parts List

The summarized parts list combines the summarized bills of material for all final products or master schedule items. The gross requirement for any final product is the total planned order quantity for that final product. The net requirement is calculated as the gross requirement minus the quantity on hand.

The summarized parts list is given by the SUMMARYOUT= output data set in PROC BOM. For details about this data set, see the section "Summarized Parts Data Set" on page 37. Figure A.4 shows the summarized parts list for the BOM Web example.

Figure A.4 Summarized Parts List

Summarized Parts List
across all planned orders

Part Number	Low-Level Code	Gross Req.	Qty. On Hand	Net Req.	Part Description	Unit	Lot Size	Type	Unit Cost
5640	4	1392.5	0	1392.5	Steel	Lbs	2500	2	115
5746	2	1250	0	1250	Hose	In	2400	2	45
4315	2	375	0	375	Brush Assembly	Ea	600	2	77
1201	3	250	0	250	Gasket	Ea	360	2	3
0403	3	125	0	125	Clamp	Ea	200	2	22
4209	3	125	0	125	Painted Tank Top	Ea	200	1	18
1959	2	125	0	125	Instruct Set	Ea	200	2	8
2156	2	125	0	125	Trigger Assembly	Ea	180	2	78
3215	2	125	0	125	Solution Tank	Ea	200	2	65
3219	2	125	0	125	Handle	Ea	200	2	89
5319	2	125	0	125	Valve Assembly	Ea	200	2	137
6221-1	2	125	0	125	Bottled Concentrate	Ea	300	2	150
7114	2	125	0	125	Power Cord	Ea	300	2	23
1115	1	125	0	125	Customer Pack	Ea	200	1	6
4651	1	125	0	125	Handle Assembly	Ea	200	1	12
1196	2	85	0	85	S-Housing	Ea	120	1	23
5705	3	45	0	45	4-Gal Painted Tank Bottom	Ea	60	1	37

Indented BOM

To reduce the size of the displays, the indented bill of material can be separated into a bill of material for each final product. Chapter 4 describes a macro, %BOMRMLB, that constructs an indented bill of material for a selected item.

Figure A.5 shows the tabular view of the Indented BOM for the 5-Gal Carpet Cleaner. In the BOM Web example, each part number is selectable and produces the Indented BOM for the item identified by the selected part number. You can also selectively collapse or expand sections of the Indented BOM by clicking on the - and + icons next to the part numbers.

Figure A.5 Indented BOM, Tabular View

Indented BOM for Product 0125

5-Gal Carpet Cleaner

Ind. Level	Part Number	Part Description	Type	Unit	Qty. Per Assembly	Unit Cost	Lead Time	Total Lead Time
0	⊟ 0125	5-Gal Carpet Cleaner	3	Ea	.	9	7	7
1	⊟ 1115	Customer Pack	1	Ea	1	6	7	14
2	– 1959	Instruct Set	2	Ea	1	8	7	21
2	– 6221-1	Bottled Concentrate	2	Ea	1	150	105	119
1	⊟ 2104	5-Gal Tank Assembly	1	Ea	1	30	14	21
2	– 5319	Valve Assembly	2	Ea	1	137	91	112
2	– 5746	Hose	2	In	10	45	28	49
2	⊟ 7350	5-Gal Tank Subassembly	1	Ea	1	25	14	35
3	– 0403	Clamp	2	Ea	1	22	14	49
3	– 1201	Gasket	2	Ea	2	3	7	42
3	⊟ 4209	Painted Tank Top	1	Ea	1	18	14	49
4	– 5640	Steel	2	Lbs	2	115	70	119
3	⊟ 5706	5-Gal Painted Tank Bottom	1	Ea	1	38	21	56
4	– 5640	Steel	2	Lbs	3	115	70	126
1	⊟ 3824	2-Hp Housing Assembly	1	Ea	1	17	14	21
2	⊟ 1198	M-Housing	1	Ea	1	25	14	35

Figure A.6 shows the diagram view of the Indented BOM for the 5-Gal Carpet Cleaner. Each node in the diagram represents a record in the tabular view as displayed in Figure A.5. This diagram is generated using the NETDRAW procedure; Example 3.1 in Chapter 3 illustrates the use of PROC NETDRAW to draw a tree diagram for an indented bill of material. In addition, you can use the WEB= option in PROC NETDRAW to construct a "clickable" Indented BOM diagram. For example, clicking on the node "2104" in Figure A.6 produces the Indented BOM shown in Figure A.7.

Figure A.6 Indented BOM, Diagram View

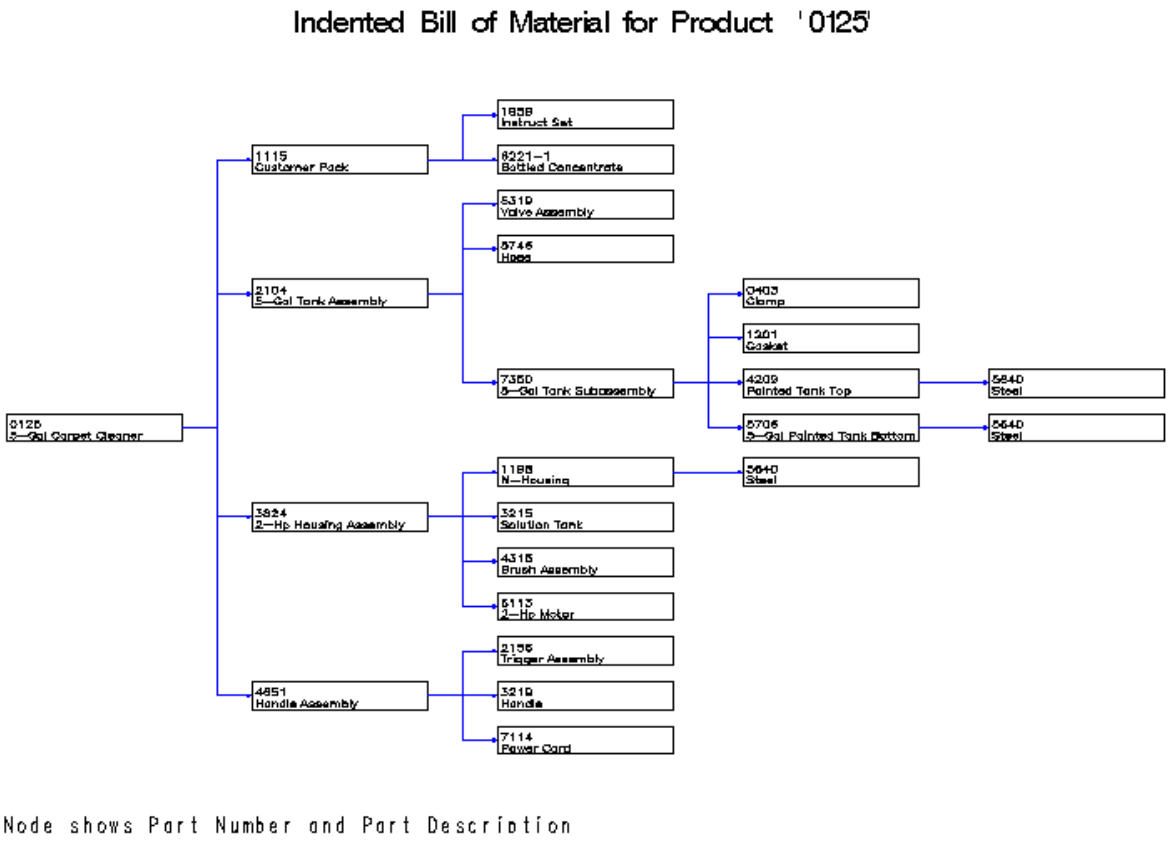
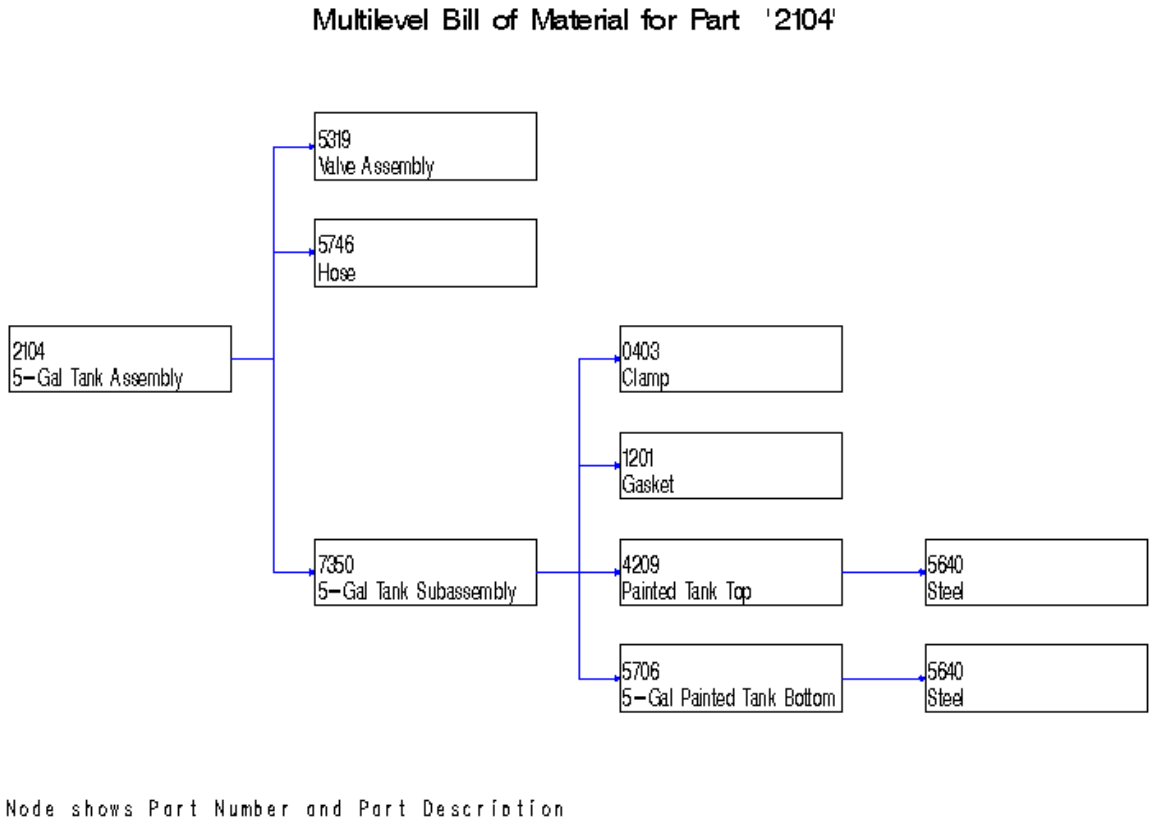


Figure A.7 Indented BOM, Diagram View Drilldown



Miscellaneous Reports

In addition to the Summarized Parts List and Indented BOM reports described in the section "Product Structure Reports" on page 129, the output from PROC BOM and the BOM reporting macros can also be used to generate summarized bills of material, gross requirement summaries, and where-used lists.

Summarized BOM

The Summarized BOM for an item describes all the parts and their total quantity required in the production of the item. For each component, the Summarized BOM lists the following information:

- Part number and description for the component
- Quantity required to make one unit of the end item
- Item type
- Unit of measure
- Unit cost
- Lead time

The Summarized BOM for the 5-Gal Tank Assembly in the BOM Web example is shown in Figure A.8. Chapter 4 describes a macro, %BOMRSUB, that constructs a summarized bill of material for a selected item.

Figure A.8 Summarized BOM

Summarized BOM for Part 2104

5-Gal Tank Assembly
Quantity Required: 1

Component	Component Description	Qty. Required	Type	Unit	Unit Cost	Lead Time
0403	Clamp	1	2	Ea	22	14
1201	Gasket	2	2	Ea	3	7
4209	Painted Tank Top	1	1	Ea	18	14
5319	Valve Assembly	1	2	Ea	137	91
5640	Steel	5	2	Lbs	115	70
5706	5-Gal Painted Tank Bottom	1	1	Ea	38	21
5746	Hose	10	2	In	45	28
7350	5-Gal Tank Subassembly	1	1	Ea	25	14

Gross Requirement Summary

The Gross Requirement Summary for an item lists the quantities of all components needed to produce the required quantity of the item. This report enables you to analyze how future orders for an item will impact inventory levels. See the section "Summarized Parts Data Set" on page 37 and Example 3.9 in Chapter 3 for details about the gross requirement calculation.

For each component, the Gross Requirement Summary includes the following:

- Part number and description for the component
- Quantity required
- Quantity on hand
- Item type
- Unit of measure
- Unit cost
- Lead time

Figure A.9 shows the Gross Requirement Summary report for the 5-Gal Tank Assembly with a required quantity of 40. Notice that this report is similar to the Summarized BOM in Figure A.8, with a column "Gross Req." that is 40 times the "Qty. Required" column from the Summarized BOM. The Gross Requirement Summary also includes a column for quantity on hand for inventory status information to compare with component requirements.

Figure A.9 Gross Requirement Summary

Gross Requirement Summary for Part 2104

5-Gal Tank Assembly
Quantity Required: 40

Component	Component Description	Gross Req.	Qty. On Hand	Type	Unit	Unit Cost	Lead Time
0403	Clamp	40	0	2	Ea	22	14
1201	Gasket	80	0	2	Ea	3	7
4209	Painted Tank Top	40	0	1	Ea	18	14
5319	Valve Assembly	40	0	2	Ea	137	91
5640	Steel	200	0	2	Lbs	115	70
5706	5-Gal Painted Tank Bottom	40	0	1	Ea	38	21
5746	Hose	400	0	2	In	45	28
7350	5-Gal Tank Subassembly	40	0	1	Ea	25	14

Where-Used Reports

The Single-Level Where-Used report for an item lists each parent in which that item is directly used and in what quantity. For each parent item, the Single-Level Where-Used report lists the following:

- Part number and description for the parent item
- Quantity required
- Item type
- Unit of measure
- Unit cost
- Lead time

The Single-Level Where-Used report for the Clamp is shown in Figure A.10. Chapter 4 describes a macro, %BOMRSLW, that constructs a single-level where-used list for a selected item.

Figure A.10 Single-Level Where-Used Report

Single-Level Where-Used List for Part 0403

Clamp

Final Product	Parent Item	Parent Description	Qty. Required	Type	Unit	Unit Cost	Lead Time
0123	1910	3-Gal Tank Subassembly	1	1	Ea	28	14
0124	4721	4-Gal Tank Subassembly	1	1	Ea	25	14
0125	7350	5-Gal Tank Subassembly	1	1	Ea	25	14

The Indented Where-Used report for an item traces its use through the indented bill of material. This report lists the item and steps through each higher-level component or assembly using the item until a final product is reached. At each level, for the current part or assembly, the Indented Where-Used report lists the following:

- Indentation level
- Part number for the parent item
- Part number and description
- Quantity required
- Item type
- Unit of measure
- Unit cost
- Lead time and total lead time
- Final product

The Indented Where-Used report for the Clamp is shown in Figure A.11. Chapter 4 describes a macro, %BOMRMLW, that constructs an indented where-used list for a selected item.

Figure A.11 Indented Where-Used Report

Indented Where-Used List for Part 0403
Clamp

Ind. Level	Parent	Part Number	Part Description	Qty. Req.	Type	Unit	Unit Cost	Lead Time	Total Lead Time	Final Product
3	1910	0403	Clamp	1	2	Ea	22	14	49	0123
2	2927	1910	3-Gal Tank Subassembly	1	1	Ea	28	14	35	0123
1	0123	2927	3-Gal Tank Assembly	1	1	Ea	23	14	21	0123
0		0123	3-Gal Carpet Cleaner	.	3	Ea	8	7	7	0123
3	4721	0403	Clamp	1	2	Ea	22	14	49	0124
2	1527	4721	4-Gal Tank Subassembly	1	1	Ea	25	14	35	0124
1	0124	1527	4-Gal Tank Assembly	1	1	Ea	28	14	21	0124
0		0124	4-Gal Carpet Cleaner	.	3	Ea	10	7	7	0124
3	7350	0403	Clamp	1	2	Ea	22	14	49	0125
2	2104	7350	5-Gal Tank Subassembly	1	1	Ea	25	14	35	0125
1	0125	2104	5-Gal Tank Assembly	1	1	Ea	30	14	21	0125
0		0125	5-Gal Carpet Cleaner	.	3	Ea	9	7	7	0125

In addition, the %BOMRSUW macro can be used to construct a summarized where-used list for a selected item, which lists the total usage of this item for every higher-level parent using it. See Chapter 4 for more information on these BOM reporting macros.

Shortage and Critical Path Analysis Reports

The BOM Web example includes reports on anticipated part shortages and analyzes the critical path for each planned order. The reports include a rough-cut shortage analysis, critical path identification, and material requirement/availability schedule. These reports are described in the following sections.

Rough-Cut Shortage Analysis Report

The Rough-Cut Shortage Analysis report describes the shortages related to the planned order. This report is termed a "rough-cut" because it does not account for on-hand stock of any items, and assumes that procurement and production begins on the user-specified start date. This report is structured according to the product's bill of materials, and for the final product and each subassembly or component it lists the following:

- Indentation level
- Part number and description

- Need date (date on which the item is needed)
- Promised date (date on which the item will be available)
- Variance (promised date minus need date; the amount of delay in availability of the item)
- Slack to next assembly (the maximum delay in availability of the item that will not delay availability of the parent item)
- Slack to final product (the maximum delay in availability of the item that will not delay receipt of the final product)
- Lead time and total lead time
- Item type
- Unit of measure
- Part number for the parent item
- Quantity per assembly of the parent item (defining the parent-component relationship)
- Final product's number
- Quantity needed to produce the amount ordered of the final product

Figure A.12 shows a portion of the Rough-Cut Shortage Analysis report for the order of 40 5-Gal Carpet Cleaners that is due on September 27, 2001. The starting date of the planning period is June 11, 2001. This figure shows only the first eight columns of the report; the remaining columns include Lead Time, Total Lead Time, Type, Unit, Parent, Qty. Per Assembly, Final Product, and Qty. Needed.

Figure A.12 Rough-Cut Shortage Analysis

Rough-Cut Shortage Analysis

Part: 0125 (5-Gal Carpet Cleaner) **Order Quantity:** 40
Start Date: 11JUN2001 **Due Date:** 27SEP2001

Ind. Level	Part Number	Part Description	Need Date	Promised Date	Variance	Slack to Next Assembly	Slack to Final Product	Lead Time
0	0125	5-Gal Carpet Cleaner	26SEP2001	14OCT2001	18	0	0	
1	1115	Customer Pack	19SEP2001	30SEP2001	11	7	7	
2	1959	Instruct Set	12SEP2001	17JUN2001	-87	98	105	
2	6221-1	Bottled Concentrate	12SEP2001	23SEP2001	11	0	7	1
1	2104	5-Gal Tank Assembly	19SEP2001	07OCT2001	18	0	0	
2	5319	Valve Assembly	05SEP2001	09SEP2001	4	14	14	
2	5746	Hose	05SEP2001	08JUL2001	-59	77	77	
2	7350	5-Gal Tank Subassembly	05SEP2001	23SEP2001	18	0	0	
3	0403	Clamp	22AUG2001	24JUN2001	-59	77	77	
3	1201	Gasket	22AUG2001	17JUN2001	-66	84	84	

Items with positive Variance will not be available in the desired quantity when needed, and hence will have shortage or schedule problems. There might be many items with such problems, but typically only a few affect the release dates for the planned orders.

You can evaluate the impact of the Variance value for an item by comparing it to the two reported slack values. If Variance does not exceed Slack to Next Assembly, availability for the item will not delay production of the listed parent component. If Variance does not exceed Slack to Final Product, availability for the item will not delay release of the order.

The items controlling their parent items' availability are those items with a zero value for Slack to Next Assembly; these are referred to as *pacing items*. Every subassembly has its own pacing item. Items with a zero value for Slack to Final Product are said to be *critical*. For example, Figure A.12 indicates that Part Numbers 0125, 6221-1, 2104, and 7350 are pacing items. In addition, Part Numbers 0125, 2104, and 7350 are critical. The Rough-Cut Shortage Analysis report uses the CPM procedure to compute the promised date, the slack to next assembly, and the slack to final product, and backward scheduling to determine the need date. You can construct a similar report with a single call to PROC CPM, followed by one data step. Suppose the indented bill of materials for the 5-Gal Carpet Cleaner is stored in a data set named IndBOM_0125. The following code creates a Rough-Cut Shortage Analysis report for the product, which

is saved in a data set named RoughCut0125. This data set contains the information listed in Figure A.12, along with some additional variables that will be used in the section "Material Requirement/Availability Schedule" on page 141 to produce Gantt charts of the schedules.

```
proc cpm data=IndBOM_0125 out=RoughCut0125
         date='11Jun2001'd addact;
   act Part_ID;
   succ Paren_ID;
   dur Leadtime;
   id _all_;
   run;

data RoughCut0125;
   format PromDate NeedDate date9.;
   set RoughCut0125 (rename= (e_finish=PromDate
                              f_float=Slack2NA
                              t_float=Slack2FP));
   NeedDate = '27Sep2001'd - (Tot_Lead-Leadtime) - 1;
   Variance = PromDate - NeedDate;
   QtyNeeded = 40*Qty_Prod;
   run;
```

Critical Path Identification Report

The critical items (items with zero Slack to Final Product) in the Rough-Cut Shortage Analysis form a path in the indented bill of material. This path is referred to as the *critical path* for the planned order. The Critical Path Identification Report lists this critical path. Among those critical items are the items that delay product availability; the ones with lowest level are most crucial and often merit special attention and expedited scheduling.

Figure A.13 shows a portion of the Critical Paths report for the order of 40 5-Gal Carpet Cleaners that is due on September 27, 2001. The figure shows only the first nine columns of the report; the remaining columns include Total Lead Time, Type, Unit, Parent, Qty. Per Assembly, Final Product, and Qty. Needed. In this example, steel (Part 5640) is the lowest level item and, therefore, is the most crucial of the critical items.

Figure A.13 Critical Path Identification Report

Critical Paths

Part: 0125 (5-Gal Carpet Cleaner) **Order Quantity:** 40
Start Date: 11JUN2001 **Due Date:** 27SEP2001

Ind. Level	Part Number	Part Description	Need Date	Promised Date	Variance	Slack to Next Assembly	Slack to Final Product	Lead Time
0	0125	5-Gal Carpet Cleaner	26SEP2001	14OCT2001	18	0	0	7
1	2104	5-Gal Tank Assembly	19SEP2001	07OCT2001	18	0	0	14
2	7350	5-Gal Tank Subassembly	05SEP2001	23SEP2001	18	0	0	14
3	5706	5-Gal Painted Tank Bottom	22AUG2001	09SEP2001	18	0	0	21
4	5640	Steel	01AUG2001	19AUG2001	18	0	0	70

Material Requirement/Availability Schedule

The Material Requirement/Availability Schedule is a Gantt chart, a time-scaled graphical depiction of the schedule for a planned order. The schedules for all items needed to fulfill the order are displayed hierarchically, with critical items depicted in red and all others depicted in green schedule bars. Schedule bars indicate the start and finish of production or procurement for each item, and also indicate any slack time that exists. Links between the schedule bars indicate the hierarchical relationships between the items. Milestone markers indicate the Need Date and Promised Date for each item, and a vertical line indicates the Due Date for the order.

The Material Requirement/Availability Schedule for the order of 40 5-Gal Carpet Cleaners is shown in Figure A.14. In this printed version, the red and green schedule bars are indistinguishable; however, the BOM Web example shows a full-color version of the Material Requirement/Availability Schedule, in which the critical items are depicted in red schedule bars, and all other schedule bars are colored green.

In the BOM Web example, the schedule bars are selectable and produce a "Material Requirement Implosion," which uses a Gantt chart to depict a full level pegging for the selected item. This is a time-scaled Indented Where-Used report for the selected item in the designated final product. Figure A.15 shows the Material Requirement Implosion produced by selecting Steel (Part 5640) from Figure A.14.

Figure A.14 Material Requirement/Availability Schedule

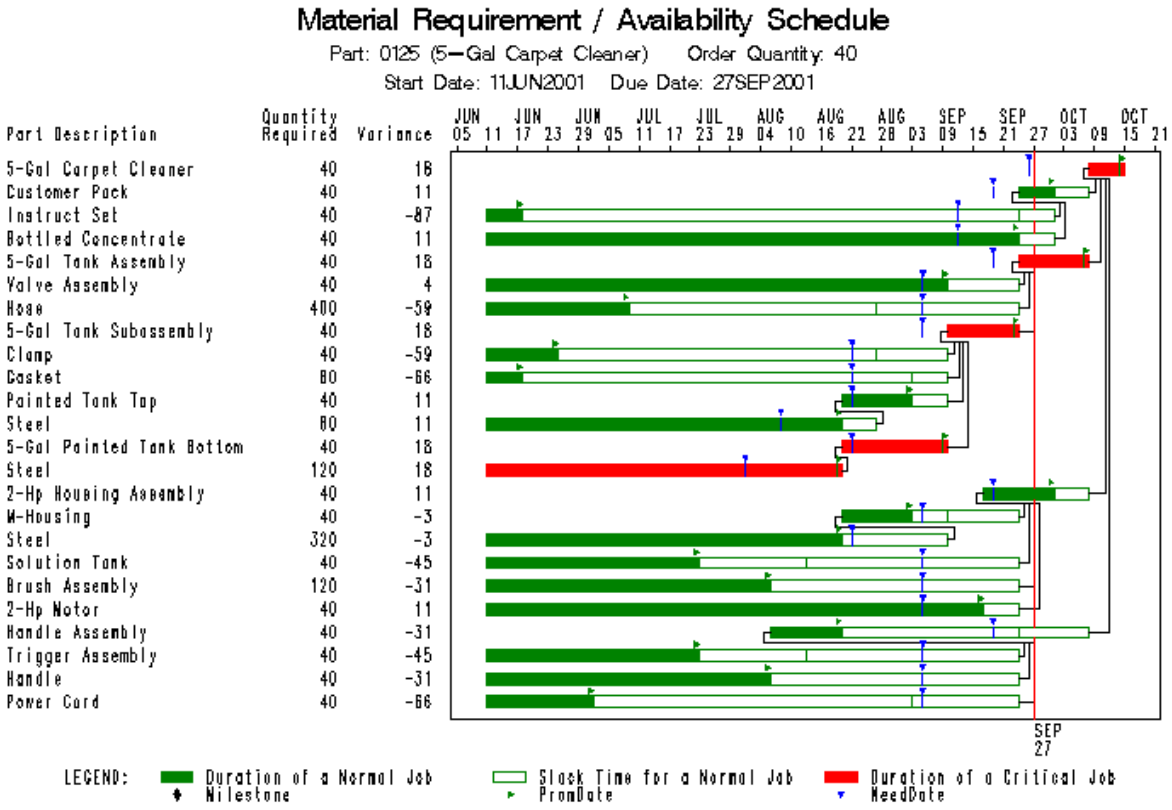
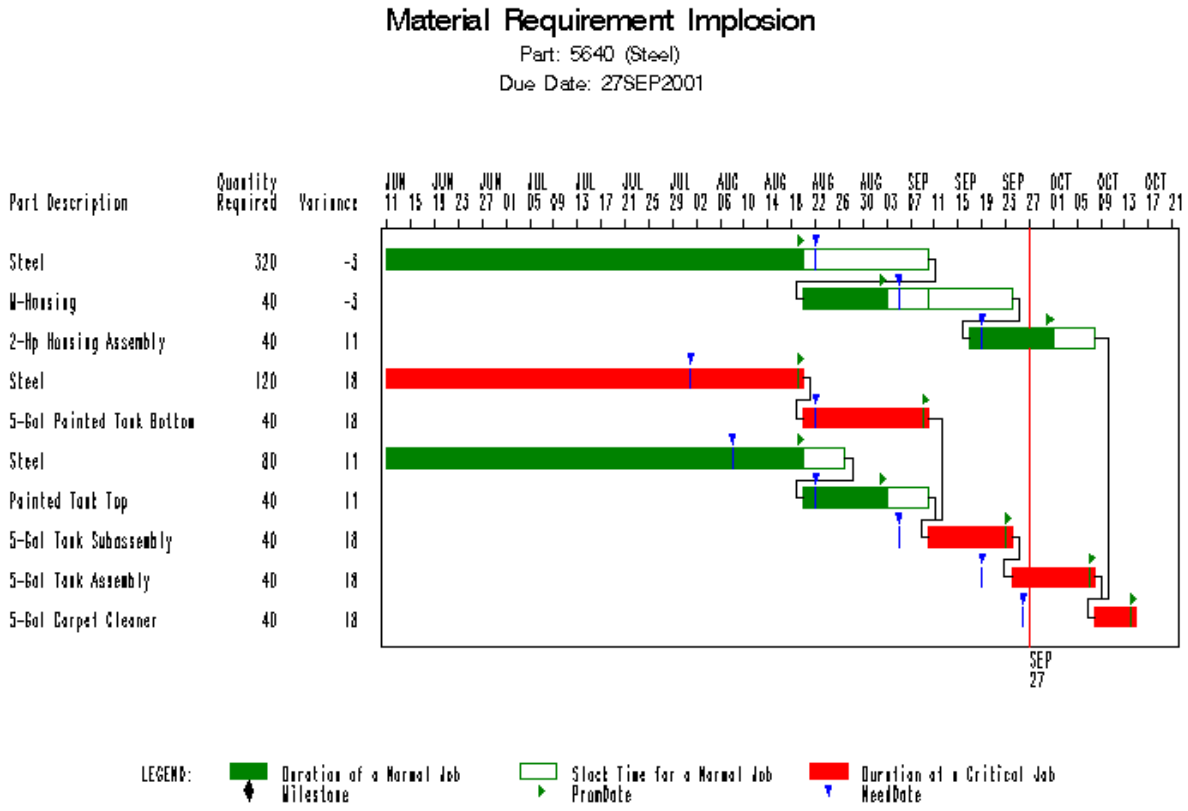


Figure A.15 Material Requirement Implosion



You can use PROC GANTT to produce Material Requirement/Availability Schedules similar to the Gantt chart shown in Figure A.14. The following code produces a Material Requirement/Availability Schedule for the 5-Gal Carpet Cleaner with an order quantity of 40, a starting date of June 11, 2001, and a due date of September 27, 2001. The code uses the RoughCut0125 data set created in the section "Rough-Cut Shortage Analysis Report" on page 137.

```
title1 'Material Requirement / Availability Schedule';
title2 'Part: 0125 (5-Gal Carpet Cleaner)    Order Quantity: 40';
title3 ' Start Date: 11JUN2001    Due Date: 27SEP2001';

pattern1 v=s c=green;
pattern2 v=e c=green;
pattern3 v=s c=red;
pattern4 v=e c=red;
symbol1  v="B" c=green f=ORFONT h=0.5;
symbol2  v="D" c=blue f=ORFONT h=0.5;


proc gantt data=RoughCut0125
           logic=RoughCut0125;
   chart PromDate NeedDate /
      act=Part_ID
      succ=Paren_ID
      duration=Leadtime
```

```
   e_finish=PromDate
   ref='27SEP2001'd
   cref=red
   reflabel
   nojobnum
   noarrowhead
   compress;
id Description Qty_Prod Variance;
label Description = 'Part Description'
      Qty_Prod = 'Quantity Required';
run;
```

Subject Index

Syntax Index

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **yourturn@sas.com**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **suggest@sas.com**.