Chapter 1
The BOM Procedure

Chapter Contents

OVERVIEW 17

GETTING STARTED 18

SYNTAX 24
- Functional Summary 24
- PROC BOM Statement 25
- STRUCTURE Statement 26

DETAILS 30
- Part Master Data Set 30
- Product Structure Data Set 32
- Indented BOM Data Set 34
- Bill of Material Explosion and Implosion 38
- Summarized Parts Data Set 40
- Missing Values in the Input Data Sets 42
- Macro Variable _ORBOM_ 43
- Computer Resource Requirements 44

EXAMPLES 44
- Example 1.1. Bill of Material with Single Input Data Set 44
- Example 1.2. Bill of Material with Lead Time Information 48
- Example 1.3. Bill of Material with Scrap Factor Information 54
- Example 1.4. Planning Bill of Material 57
- Example 1.5. Modular Bills of Material 63
- Example 1.6. Bills of Material with Repeated Relationships 66
- Example 1.7. Bills of Material Verification 70
- Example 1.8. Roll-Up Cost in Indented Bill of Material 73
- Example 1.9. Bill of Material Explosion 76
- Example 1.10. Aggregating Forecasts Using PROC BOM 79
- Statement and Option Cross-Reference Tables 83

REFERENCES 84

Chapter 1
The BOM Procedure

Overview

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 bills 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 bills 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 bills 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 1.1 on page 44 and Example 1.2 on page 48 as examples of using the NETDRAW procedure to draw a tree diagram for an indented bill of material. Chapter 2, "Bill-of-Material Post-Processing 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 bills 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

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 1.1, 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 1.2, 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 1.2), the value `'1100'` for the Parent variable is linked to the first observa-

tion of the data set PMaster0 (as displayed in Figure 1.1), 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.

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 1.1. Part Master File (PMaster0)

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

Figure 1.2. Product Structure File (ParComp0)

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 1.3. 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.

ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Desc	Qty_Per	Qty_Per	Unit	Par_ID	Par_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

Figure 1.3. Indented Bill of Material (IndBOM0)

As discussed in the "Overview" section beginning on page 17, 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 1.4 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.

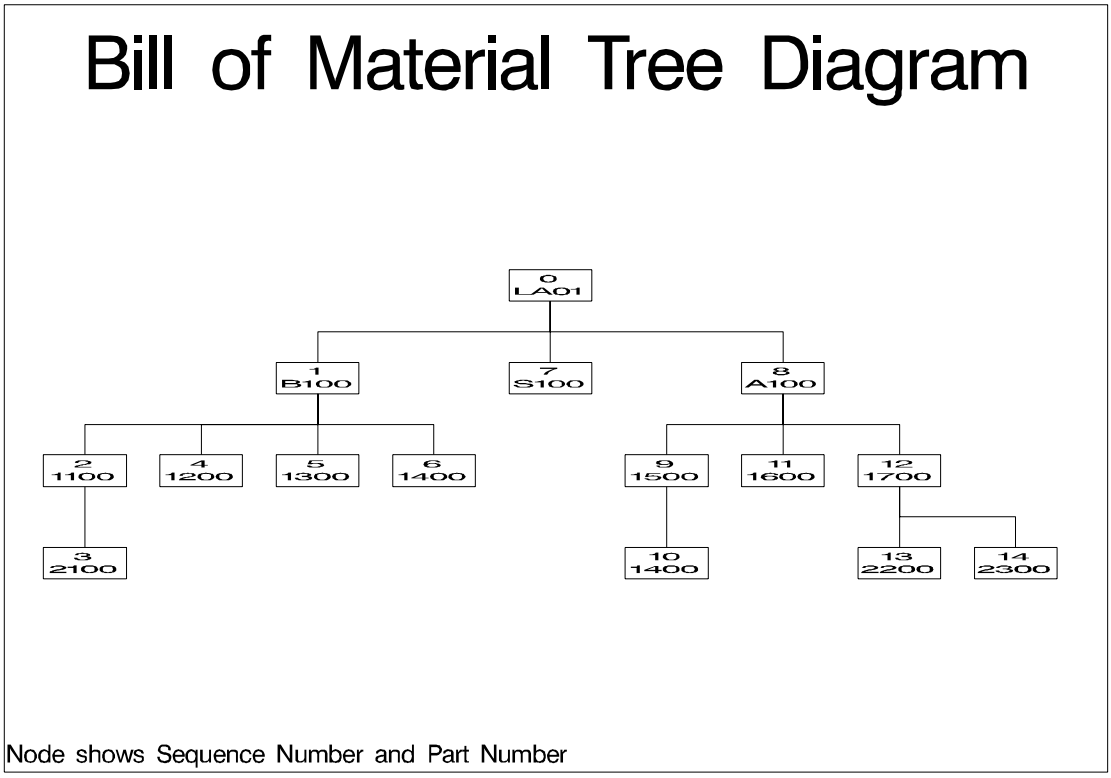


Figure 1.4. Tree Diagram for the Bill of Material for LA01

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 1.3, as well as Figure 1.4, 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 on depth-first ordering. See also the "Indented BOM Data Set" section beginning on page 34 for details regarding the records in this data set.

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

Figure 1.5. Summarized Parts List (SumBOM0)

The Summarized Parts data set, SumBOM0, is displayed in Figure 1.5. 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 1.1) 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 "Summarized Parts Data Set" section beginning on page 40 for details on determining the gross and net requirements.

Recall that the Part Master data set (as shown in Figure 1.1) 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 1.5 on page 22) 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

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 1.1. PROC BOM Options

Task	Statement	Option
Data Set Specification		
Product Structure input data set	PROC BOM	DATA=
Part Master input data set	PROC BOM	PMDATA=
Indented BOM output data set	PROC BOM	OUT=
Summarized Parts output data set	PROC BOM	SUMMARYOUT=
Part Master Information Specification		
ID *variables*	STRUCTURE	ID=
lead time *variable*	STRUCTURE	LEADTIME=
part *variable*	STRUCTURE	PART=
quantity on hand *variable*	STRUCTURE	QTYONHAND=
gross requirement *variable*	STRUCTURE	REQUIREMENT=
Problem Size Options		
do not use utility data set	PROC BOM	NOUTIL
number of items	PROC BOM	NPARTS=
number of product structure records	PROC BOM	NRELTS=
Product Structure Specification		
end items	STRUCTURE	ENDITEM=
Identical Relationships Handling Specification		
handling of identical product structure records	PROC BOM	DUPLICATE=
Relationship Information Specification		
component *variables*	STRUCTURE	COMPONENT=
scrap factor *variables*	STRUCTURE	FACTOR=
lead-time offset *variables*	STRUCTURE	OFFSET=
parent *variable*	STRUCTURE	PARENT=
quantity per assembly *variables*	STRUCTURE	QUANTITY=
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, etc.) 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 "Product Structure Data Set" section on page 32 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 "Product Structure Data Set" section beginning on page 32 for details on 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 "Computer Resource Requirements" section on page 44 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 "Computer Resource Requirements" section on page 44 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 bills 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 "Indented BOM Data Set" section on page 34 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 "Part Master Data Set" section on page 30 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 "Summarized Parts Data Set" section on page 40 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 bills of material. In other words, this option can be used to construct indented bills of material and summarized parts list for subassemblies. 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 ith factor variable corresponds to the ith 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 "Indented BOM Data Set" section on page 34 for details on 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 "Indented BOM Data Set" section on page 34 for details on 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 ith quantity variable corresponds to the ith 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 "Summarized Parts Data Set" section beginning on page 40 for details on the

calculation of the gross requirements.

RID=(*variables*)
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

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 "Summarized Parts Data Set" section beginning on page 40 for details on 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 1.2 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 1.2. 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
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 1.1.1 on page 45 and the SIBOM2 data set shown in Output 1.2.1 on page 49 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 1.1.1 on page 45 as an example. You can also include part master data in this data set. See Example 1.1 on page 44 and Example 1.2 on page 48 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 1.3 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 1.3. 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 1.6 on page 66 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 bills 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 "Getting Started" section beginning on page 18, 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 1.4 lists all the variables in the Indented BOM data set. It also lists the type and a brief description of these variables.

Table 1.4. 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
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 bills 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 1.3 on page 20, 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 1.3 on page 20 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 "Bill of Material Explosion and Implosion" section beginning on page 38 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 1.3 on page 20. The single-level bill of material is shown in Figure 1.6.

```
  /* 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;
```

```
                        ABC Lamp Company

          Single-level Bill of Material Retrieval, Part LA01

                                          Qty
     Component          Desc             Per    Unit

       B100        Base assembly          1     Each
       S100        Black shade            1     Each
       A100        Socket assembly        1     Each
```

Figure 1.6. Components Directly Used in Part LA01

The single-level where-used list for item `'1400'`, displayed in Figure 1.7, 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;
```

```
                        ABC Lamp Company

          Single-level Where-used Report, Part 1400

                                          Qty
     _Parent_           Desc             Per    Unit

       B100        Base assembly          4     Each
       1500        Steel holder           2     Each
```

Figure 1.7. Parents in which Part 1400 is Directly Used

The "Reporting Macros" section (beginning on page 91) in Chapter 2, "Bill-of-Material Post-Processing Macros," describes six SAS autocall macros that can be used to create single-level, indented, and summarized bills of material and where-used lists for an item from the Indented BOM data set. Example 1.8 on page 73 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 1.9 on page 76 and Example 1.10 on page 79 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 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

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

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

Table 1.5. 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 1.6. 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
	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

Table 1.6. (continued)

Data Set	Variable	Value Used/Assumption Made/Action Taken
	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 1.1.1 on page 45 and Output 1.3.2 on page 55 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 "Summarized Parts Data Set" section on page 40 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 "PROC BOM Statement" section (beginning on page 25) 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

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 "Getting Started" section beginning on page 18.

Example 1.1. 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 1.1.1. 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 1.1.1. 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 1.1.2, and the summarized parts list, which has been sorted by the _Part_ variable, is displayed in Output 1.1.3. These are exactly the same as the indented bill of material shown in Figure 1.3 on page 20 and the summarized parts list shown in Figure 1.5 on page 22.

```
  /* 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 1.1.2. Indented Bill of Material (IndBOM1)

ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Desc	_Qty_	Qty_Per	Unit	Par_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 1.1.3. 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

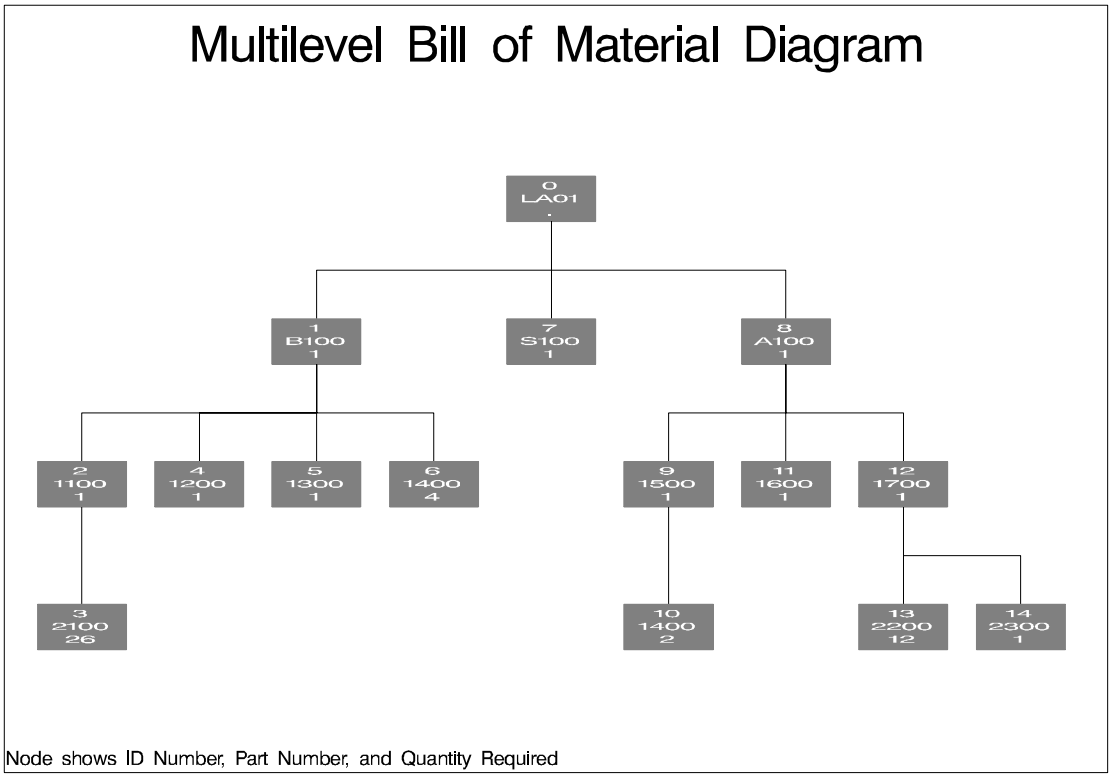
As discussed in the "Summarized Parts Data Set" section beginning on page 40, since item 'LA01' is the only master schedule item in this example, and both the quantity on hand and the scrap factor are zero for all items, the summarized parts list displayed in Output 1.1.3 is also the summarized bill 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 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 1.1.4. For further details about the NETDRAW procedure, refer to Chapter 5, "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-Component 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=white;

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 .) );
   actnet / act=Paren_ID succ=Part_ID id=(Paren_ID _Part_ QtyPer)
            ctext=black font=swiss htext=3 carcs=black
            ybetween=3 xbetween=8 centerid
            tree pcompress rotatetext rotate
            arrowhead=0 rectilinear nodefid nolabel;
run;
```

Output 1.1.4. Tree Diagram for the Bill of Material



Example 1.2. Bill of Material with Lead Time Information

As in Example 1.1 on page 44, 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 1.1.1 on page 45) in which each observation contains three product structure records, each observation in the input data set SIBOM2, as depicted in Output 1.2.1, 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 1.2.1. 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 1.2.2. 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 1.2.2. 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	LT	Total_LT	_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 1.2.3. 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 1.2.3 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;
```

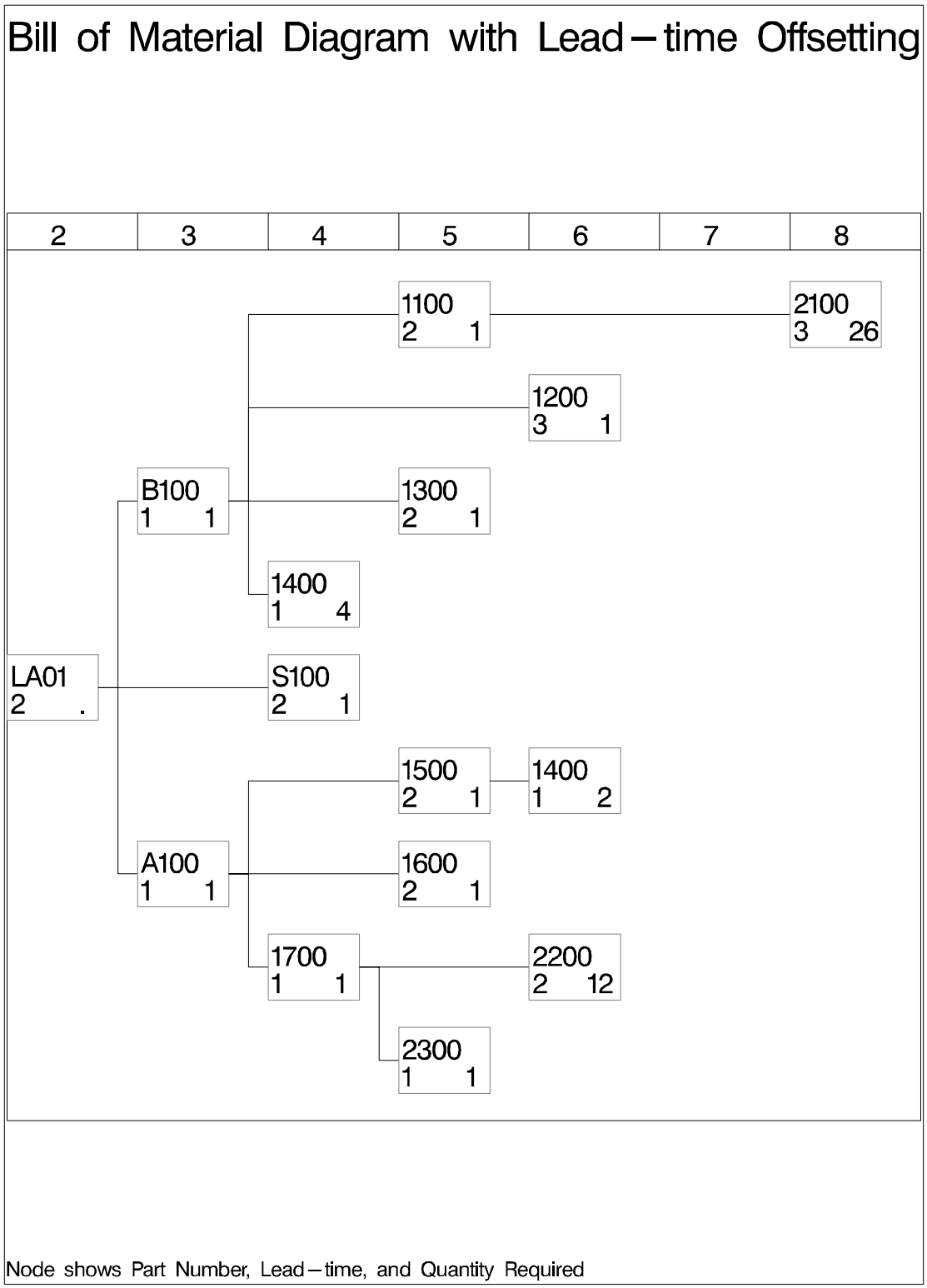
PROC NETDRAW is invoked with the NODISPLAY option to create a data set (Layout2) that contains all the layout information for the tree structure. The OUT= option specifies the name of the layout data set:

```
  /* Specify graphics options */
title h=1 j=c 'Bill of Material Diagram with Lead-time Offsetting';
footnote h=0.5 j=l
         'Node shows Part Number, Lead-time, and Quantity Required';
pattern1 v=e c=gray;

  /* 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 1.2.4. Refer to Chapter 5, "The NETDRAW Procedure," (*SAS/OR User's Guide: Project Management*) for details on 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;
   actnet / id=(_Part_ LTnQP)
            font=swiss ctext=black htext=3 carcs=black
            align=Tot_Lead frame pcompress
            xbetween=15 ybetween=3
            arrowhead=0 rectilinear nodefid nolabel;
run;
```

Output 1.2.4. Bill of Material Diagram with Lead-time Offsetting



Example 1.3. Bill of Material with Scrap Factor Information

As in the introductory example described in the "Getting Started" section (beginning on page 18), this example uses two data files, PMaster3 and ParComp3, as input data sets for the BOM procedure. The PMaster3 data set, shown in Output 1.3.1, 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 1.3.2) 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 1.3.1. 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 1.3.2. 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 1.3.3 is similar to the one displayed in Output 1.2.2 on page 50, 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 1.3.2. 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 "Indented BOM Data Set" section on page 34 for information about determining the quantity per product when scrap factor is in effect.

Output 1.3.3. 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 1.3.4. Comparing it with the summarized parts list displayed in Output 1.1.3 on page 46, 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 1.3.4 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 1.3.4. Summarized Parts List (SumBOM3)

ABC Lamp Company

Summarized Parts List, Period 1

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 1.4. 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 ($= 8 \times 3 \times 2$) 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 1.3.1 on page 54). Finally, the ParComp4 data set contains the re-grouped product structure information.

```
   /* Generic and phantom Part Master data */
data Phantom4;
   format Part $8. Desc $24. Unit $8.;
   input Part & Desc & Req Unit & LeadTime;
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 options and alternative parts */
data Option4;
   format Part $8. Desc $24. Unit $8.;
   input Part & Desc & Req Unit & LeadTime;
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;
   format Part $8. Component $8. QtyPer 8.2 ;
   input Part $ 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
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 1.3.1 on page 54. The SAS code to accomplish this is as follows:

```
  /* Append the old Part Master data to the new   */
  /* phantom item and additional option data sets */
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 1.4.1. Output 1.4.2 lists the quantity of each item that is needed to build 10,000 lamps in period 4, sorted by the _Part_ variable.

Output 1.4.1. 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 1.4.2. 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 1.5. Modular Bills of Material

The previous example illustrated how the bills 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 bills of material*. One reason is that the task of maintaining modularized bills of material is much simpler. The only bills 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 bills of material, if necessary. This example shows how to build modular bills of material for the 48 product configurations as described in Example 1.4 (page 57). It also demonstrates how to construct a final-product-oriented bill of material from the modularized bills 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 1.4 on page 57), and the new Product Structure data set, ParComp5, to construct modular bills of material for the production line of the company. It also displays the modularized indented bills 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 1.5.1. This output data set contains a pseudo bill of material for the pseudo end item 'LAXX' and 13 modules (modular bills of material) for items 'A100', 'A101', 'B100', 'B101', 'B102', and 'S100' ..., 'S107', respectively.

Output 1.5.1. Modularized Indented Bills of Material (IndBOM5)

ABC Lamp Company

Indented Bills of Material

Level	_Parent_	_Part_	Desc	Qty Per	Qty_Prod	Unit	Lead Time	Total Lead_Time	_Prod_
0		A100	Socket assembly	.	1	Each	1	1	A100
1	A100	1600	One-way socket	1	1	Each	2	3	A100
0		A101	Three-way socket assem.	.	1	Each	1	1	A101
1	A101	1601	Three-way socket	1	1	Each	2	3	A101
0		B100	Base assembly	.	1	Each	1	1	B100
1	B100	1200	6-Diameter steel plate	1	1	Each	3	4	B100
0		B101	7in Base assembly	.	1	Each	1	1	B101
1	B101	1201	7-Diameter steel plate	1	1	Each	3	4	B101
0		B102	8in Base assembly	.	1	Each	1	1	B102
1	B102	1202	8-Diameter steel plate	1	1	Each	3	4	B102
0		LAXX	Lamp LA	.	1	Each	3	3	LAXX
1	LAXX	4000	Common parts	1	1	Each	0	3	LAXX
2	4000	1100	Finished shaft	1	1	Each	2	5	LAXX
3	1100	2100	3/8 Steel tubing	26	26	Inches	3	8	LAXX
2	4000	1300	Hub	1	1	Each	2	5	LAXX
2	4000	1400	1/4-20 Screw	4	4	Each	1	4	LAXX
2	4000	1500	Steel holder	1	1	Each	2	5	LAXX
3	1500	1400	1/4-20 Screw	2	2	Each	1	6	LAXX
2	4000	1700	Wiring assembly	1	1	Each	1	4	LAXX
3	1700	2200	16-Gauge lamp cord	12	12	Feet	2	6	LAXX
3	1700	2300	Standard plug terminal	1	1	Each	1	5	LAXX
1	LAXX	B10X	Base assembly options	1	1	Each	0	3	LAXX
1	LAXX	S10X	Shade options	1	1	Each	0	3	LAXX
1	LAXX	A10X	Socket assembly options	1	1	Each	0	3	LAXX
0		S100	Black shade	.	1	Each	2	2	S100
0		S101	14in White shade	.	1	Each	2	2	S101
0		S102	14in Cream shade	.	1	Each	2	2	S102
0		S103	14in Yellow shade	.	1	Each	2	2	S103
0		S104	15in Black shade	.	1	Each	2	2	S104
0		S105	15in White shade	.	1	Each	2	2	S105
0		S106	15in Cream shade	.	1	Each	2	2	S106
0		S107	15in Yellow shade	.	1	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 bills of material for 'S100' and 'A100'. Note that the %BOMTSAE macro refers to the part master file PMaster3 (shown in Output 1.3.1 on page 54) for the master data of the item 'LA01'. See the "Transactional Macros" section beginning on page 111 for details on 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 1.5.2. It is the same as the bill of material displayed in Output 1.2.2 on page 50 (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 1.5.2. Indented Bill of Material for Product 'LA01' (BomOut5)

ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Desc	Qty_Per	Qty_Prod	Unit	Lead_Time	TotLead	_Prod_
0		LA01	Lamp LA	.	1	Each	2	2	LA01
1	LA01	4000	Common parts	1	1	Each	0	2	LA01
2	4000	1100	Finished shaft	1	1	Each	2	4	LA01
3	1100	2100	3/8 Steel tubing	26	26	Inches	3	7	LA01
2	4000	1300	Hub	1	1	Each	2	4	LA01
2	4000	1400	1/4-20 Screw	4	4	Each	1	3	LA01
2	4000	1500	Steel holder	1	1	Each	2	4	LA01
3	1500	1400	1/4-20 Screw	2	2	Each	1	5	LA01
2	4000	1700	Wiring assembly	1	1	Each	1	3	LA01
3	1700	2200	16-Gauge lamp cord	12	12	Feet	2	5	LA01
3	1700	2300	Standard plug terminal	1	1	Each	1	4	LA01
1	LA01	A100	Socket assembly	1	1	Each	1	3	LA01
2	A100	1600	One-way socket	1	1	Each	2	5	LA01
1	LA01	B100	Base assembly	1	1	Each	1	3	LA01
2	B100	1200	6-Diameter steel plate	1	1	Each	3	6	LA01
1	LA01	S100	Black shade	1	1	Each	2	4	LA01

Example 1.6. Bills of Material with Repeated Relationships

As briefly described in the "Product Structure Data Set" section beginning on page 32, 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 1.6.1 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 1.6.1. 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 1.3.1 on page 54), 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 1.6.2.

```
  /* 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 1.6.2. Indented Bill of Material with Repeated Relationships

ABC Lamp Company

Indented Bill of Material, Part LA01

Obs	_Level_	_Parent_	_Part_	Desc	_Qty_	_Fscrap_	_Qty_	_Unit_	_Lead_	_To_	_Lead_	_To_	_Lead_	_To_	_Point_	_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 bills 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 bills of material. The following SAS code invokes PROC BOM to create a bill of material with identical relationships, displayed in Output 1.6.3. 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 1.6.3. Indented Bill of Material with Identical Relationships

ABC Lamp Company

Indented Bill of Material, Part LA01

_ Level _	_ Parent _	_ Part _	Desc	Qty Per	Fscrap	Qty _ Unit	Unit	Lead Time	Total Lead Time	_ 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 bills 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 1.6.4. Note that the two identical relationships of the parent `'B100'` and the component `'1400'` are collapsed into one relationship. See the "Product Structure Data Set" section beginning on page 32 for details on combining identical parent-component relationships.

Output 1.6.4. Indented Bill of Material with Identical Relationships Collapsed

ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Desc	Qty Per	Fscrap	Qty Prod	Unit	LT Time	LT Load	Top_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 1.7. Bills 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 bills 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 1.4 on page 57. 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 "Part Master Data Set" section beginning on page 30 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 1.4 on page 57) 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 1.4 on page 57). The summarized parts list is shown in Output 1.7.1.

```
   /* 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 1.7.1. 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 1.7.1 and the one in Output 1.1.3 on page 46 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 1.1.3.

Another way to compare two bills of material is to compare their summarized bills of material. Recall that, as discussed in Example 1.1, the summarized parts list as displayed in Output 1.1.3 on page 46 is also the summarized bill of material for the item `'LA01'`. You can also use the %BOMRSUB SAS macro described in Chapter 2, "Bill-of-Material Post-Processing Macros," or the SAS DATA step to create the summarized bill of material from an indented bill of material. See Example 1.9 on page 76 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

1.5.2 on page 66. 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 1.7.2. Summarized Bills 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 1.7.2, agrees with the summarized bill of material displayed in Output 1.1.3 on page 46 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 1.7.2.

Example 1.8. Roll-Up Cost in Indented Bill of Material

As mentioned in the "Bill of Material Explosion and Implosion" section beginning on page 38, 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 1.1 on page 19 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 1.2 (page 19) 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 1.8.1. 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 1.8.1. Indented Bills of Material with Roll-Up Cost (IndBOM8a)

ABC Lamp Company

Indented Bill of Material, Part LA01

Level	_Parent_	_Part_	Desc	Qty Per	Qty_Prod	Unit	Cost
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 1.9. Bill of Material Explosion

Example 1.8 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 "Summarized Parts Data Set" section beginning on page 40 for details on calculating the gross and net requirements for each item.

The following SAS code uses the data set SlBOM2 (displayed in Output 1.2.1 on page 49) 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 1.9.1, and the summarized parts list, SumBOM9, is shown in Output 1.9.2.

```
  /* 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 1.9.1. 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 1.9.2. 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 1.9.2 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 bills 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 1.9.1 to produce a gross requirements report for an additional order of 50 units of 'LA01'. The gross requirements report, SumReq9, is displayed in Output 1.9.3; 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 1.9.3. 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 1.9.4.

```
  /* 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 1.9.4. Total Usages for Making 50 Units of 'LA01'

ABC Lamp Company

Summarized Bill of Material, Part LA01: Requirement=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 1.7 on page 70, you can also use the %BOMRSUB SAS macro described in Chapter 2, "Bill-of-Material Post-Processing Macros," to create the same report as that shown in Output 1.9.4.

Example 1.10. 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 $12. 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 1.10.1, Output 1.10.2, and Output 1.10.3, 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';
   var _Part_ code dcitems;
run;
```

Output 1.10.1. 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 1.10.2. 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 1.10.3. 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 1.7. 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, Inc.

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 Co.

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

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