USPS Cost and Revenue Analysis Model
Excel/VBA Program Documentation

Cost and Revenue Analysis Model Programmer's Manual

TABLE OF CONTENTS

I. OVERVIEW 1
A. PURPOSE OF THE COST AND REVENUE ANALYSIS MODEL 1
B. BACKGROUND AND MODEL UPDATE 1

II. EXCEL/VBA MODEL DESIGN 2
A. MODEL INPUTS 2
B. PROCESSING STEPS 2
C. CONTROL TABLES 3
D. CRA MODEL ROUTINES 4
E. CRA MODEL OUTPUTS 5
F. BASIC FEATURES 6
G. HARDWARE 6
H. SOFTWARE 6

III. VBA PROGRAM OVERVIEW 13
A. Main Program (ModDriver) 13
B. Initialization Routines (ModInit) 13
C. Calculations (ModRoutines) 14
D. SubTotals (ModSubtotals) 14
E. Reporting (ModReports) 14
F. Writing Output Workbooks (ModWorkbooks) 14
G. Response to User Actions (ModUserInterface) 15
H. Utility Functions (ModUtil) 15
I. Error Handling Routines (modErrorHandling) 15

IV. FLOW DIAGRAM 17

V. VBA PROGRAM DESCRIPTION 25
A. Main Program - ModDriver 25
B. Initialization Routines (ModInit) 27
C. Calculations (ModRoutines) 41
D. SubTotals (ModSubtotals) 53
E. Reporting (ModReports) 55
F. Writing Output Workbooks (ModWorkbooks) 62
G. Response to User Actions (ModUserInterface) 68
H. Utility Functions (ModUtil) 73
I. Error Handling Routines (ModErrorHandling) 75

VI. VBA PROGRAM CODE 82
A. Main Program – ModDriver 82
B. Initialization Routines – ModInit 85
C. Calculations – ModRoutines 120
D. SubTotals – ModSubtotals 153
E. Reporting – ModReports 159
F. Writing Output Workbooks – ModWorkbooks 193
G. Response to User Actions – ModUserInterface 212
H. Utility Functions – ModUtil 223
I. Error Handling Routines – ModErrorHandling 227

VII. DIRECTORY OF GLOBAL VARIABLES 246

I. OVERVIEW

A. PURPOSE OF THE COST AND REVENUE ANALYSIS MODEL

The Cost and Revenue Analysis (CRA) model produces a set of reports on fiscal year postal costs. The CRA itemizes costs by major categories or cost segments, such as postmasters, supervisors, etc. The costs for the Postal Service are also categorized as those costs which can be attributed to a class of mail or type of mail service, and any costs which cannot be attributed

Costs are itemized by major categories or cost segments, such as postmasters, supervisors, city delivery, transportation, etc, and by individual cost elements, or components, within cost segments. The costs for the Postal Service are also categorized as those costs which can be attributed to a class of mail or type of mail service, and any costs which cannot be attributed. For each relevant cost component, the CRA model distributes postal costs to mail classes.

B. BACKGROUND AND MODEL UPDATE

The Postal Service has converted and redesigned its financial modeling system from a mainframe-based COBOL application to a PC-platform using off-the-shelf software. The driving factors for this redesign were to reduce the learning curve needed to use and run the model, to improve maintainability, to increase flexibility, and to give greater visibility to model actions. The new model has been developed with Microsoft Excel as the platform.

II. EXCEL/VBA MODEL DESIGN

A. MODEL INPUTS

Input to the CRA model comes from a variety of sources. This input consists of:

1) Costs for components that have already been distributed to mail classes (independent components)
2) Distribution keys
3) Total costs for components to be distributed to mail classes (dependent components)
4) Total costs for components that will not be distributed to mail classes
5) Factors to determine the portion of total cost in a component that is volume variable
6) The proportion of total space usage or rental value for each of 74 space and rental value categories
7) The proportion of total equipment maintenance labor, equipment parts and supplies, or equipment capital for each of 20 equipment categories.
8) Other factors as needed.
9) Control tables, described in section D.

B. PROCESSING STEPS

The CRA model processing proceeds in several steps:

1) Read all data into the input cost matrix and produce an input report showing the distributed or total cost for each input component and factor values.
2) Distribute all dependent components except plant, equipment, and selected administrative (PESSA) costs. This is called the A module, and the associated report is the A report.
3) Generate distribution keys for space usage, rental value, and equipment. This is called the F module, and the associated report is the F report.
4) Distribute PESSA components. This is called the B module, and the associated report is the B report.
5) Generate the Cost Segments and Components and Final Reports (C and D Reports).

Each module in steps 2, 3, and 4 above produces an output cost matrix, showing the values at that step in the CRA model processing for each cost component and distribution key component.

The Roll Forward model uses the output cost matrix from the A Report. After test year costs have been estimated using the Roll Forward model, then the resulting test year cost matrix output by the Roll Forward model can be input into the CRA model for production of steps 3 through 5 above.

C. CONTROL TABLES

The Excel-based CRA model is driven by a set of control tables. The Component Master List contains a line for each cost, distribution key, and factor component. Each line includes the component's name, number, documentation information, variability (if applicable), printing requirements, subtotaling requirements, and the actions by module to be performed on the component. The actions will be described in a series of columns for each module – i.e. a set of columns for the A module, a set of columns for the F module, and a set of columns for the B module. There is also a set of columns to be used for any preprocessing needs, such as the GEDI adjustment. The actions to be performed on any cost or distribution key component are limited to a set of routines described in the next section. See Table I for a sample of an extract of the Component Master List.

Distribution keys are defined in two control tables. The Distribution Key Master List shows the distribution key name and number, the components distributed by the key, whether the distribution key is comprised of the sum of one or more other existing components, and indicators whether it is used in the A, F, and/or B modules, and in the Roll Forward model. See Table II for an extract of the Distribution Key Master List.

The Distribution Key Addends control table defines the components for each distribution key that is the sum of two or more components. See Table III for an extract of the Distribution Key Addends table.

The D Report control table describes the processing steps needed to compute the final D report. The D report includes a set of routines used only in D report processing. D report processing steps are implemented at the end of the B module processing. See Table IV for the D Report input table.

The control tables have been designed to fully document CRA model processing. The Master Component List documents the input source for each component, distribution key, and factor, and also documents the USPS Library Reference or Workpaper, the input type (rate, total), and sub-segment (Summary Description) reference, and the contents of each subtotal component. The Distribution Key control tables document the contents of each distribution key, the distributed components, and the order of distribution. This documentation, along with documentation that will be printed in the actual I, A, F, B, C, and D reports replaces the footnotes in the COBOL CRA model reports and Library Reference-5.

D. CRA MODEL ROUTINES

The CRA model control table defines the CRA model action, or routine, to perform in each cost or distribution key component in each module. Table V describes the routines used in the CRA model. The routines are processed in a particular order. For each module, the CRA model will first process all of the Phase I routines, then all of the Phase I routines, then the Phase III routines, and finally, the Phase IV routines.

Phase I, III, and IV routines are processed in the order of components on the Component Master List.

Phase II includes all of the distribution routines. Phase II routines are processed in the order of distribution keys on the Distribution Key Master List. The processing for Phase II routines proceeds as follows, using the A Module as an example:

> Step 1. Go to the Distribution Key Master List, and find the first distribution key on the list that is used in the A Module.
>
> Step 2. If the distribution key contains more than one component, then calculate that distribution key. The distribution key is calculated using the definition in the Distribution Key Addends control table, by adding the costs by mail class for all of the cost components that comprise the distribution key.
>
> Step 3. Go back to the Component Master List, find each component using the distribution key in a Phase II routine, and perform the appropriate distribution routine.
>
> Step 4. Go back to the Distribution Key Master List and find the next distribution key on the list that is used in the A Module. Continue Steps 1 through 3 until the bottom of the Distribution Key Master List is reached.

After all of the routines for a given module have been processed, the subtotal components are then calculated. The Component Master List contains a column detailing the parent/child relationships for each Module, which is used to calculate component subtotals.

NOTE: Subtotal components that do not fit into the parent/child structure can be calculated by placing these components as a line on the Distribution Key Master List and indicating that component is used in the relevant Module. The subtotal component will be calculated based on its definition in the Distribution Key Addends worksheet as part of Phase II processing, even it is not used as to distribute any of the Module cost components.

Table V is a list of the CRA routines used in A, F, and B module processing. Table VI is a list of CRA routines used only in the D report processing.

E. CRA MODEL OUTPUTS

The CRA Model produces a set of workbooks that implement the routines for each module. The name of each workbook will contain the name of the CRA model run and the module name. For example, the base year 2004 USPS A module workbook will be called BY04.A.xls. Each workbook contains:

1. An input matrix, which is a 200 x 1600 table showing the values for each cost and distribution key component before implementation of the module routines.

2. A worksheet for each cost segment containing a column for each component in that cost segment. Formulas that implement the routine processing for that component are in the workbook, as well as documentation for what routine has been implemented and the distribution key component used (if relevant).

3. Distribution key components have been organized into 'cost segments'. There is a page for each distribution key 'cost segment', with the processing used to calculate that distribution key, if relevant.

4. An output matrix, which is a 200 x 1600 table showing the values for each cost and distribution key component after all processing for the module has been implemented.

The CRA Model also produces report workbooks that do not contain formulas and are reading for printing. To print each report, load the report into Excel, select all worksheets, and then print the report. The report workbooks can also be transmitted electronically.

The name of each report workbook will contain the name of the CRA model run and the report name. For example, the base year 2004 USPS A report workbook will be called BY04.ARpt.xls. The 'BY' prefix in the model run will cause the CRA model to add 'Base Year' to report titles. A 'FY' prefix in the model run wll cause the CRA model to add 'Fiscal Year' to report titles.

F. BASIC FEATURES

- PC based program
- Microsoft Excel is the platform (Visual Basic for Applications is an integrated package)
- User specified effects parameters entered by users in model and in separate workbooks called Control files.
- CRA dictated parameters specified in Model
- Calculations are performed in module calculation workbooks (model generates formulas contained in workbooks based on user specifications).
- Module calculation workbooks generated by model are not linked. A new run is required to change results

G. HARDWARE

Recommend run on PC with Pentium IV with 256 Mb of RAM

H. SOFTWARE

Recommend MS Excel 2000 or MS Excel XP

TABLE I. COMPONENT MASTER LIST EXTRACT

Row Seq.	*Com-ponent Number*	*Component Title*	*Sub-Segment Number*	*Source*	*USPS Library Reference/ Workpaper*	*Input Type*	Printed in "I" Report	*"A" Module Routines*	"A" DK or Source Component	"A" Add'l Parms	Printed in "A" Report	"A" Report Parent
		C/S 1 Postmasters										
1	0001	Postmasters EAS 23 & Below	1.1	C/S 1 Outputs to CRA Model	USPS-T-9: B-1	Rate	BOTH				BOTH	0251
2	0002	Postmasters EAS 24 & Above	1.2	C/S 1 Outputs to CRA Model	USPS-T-9: B-1	Total	BOTH				BOTH	0251
3	**0251**	***C/S 1 Postmasters***		Calculated in Model							CS	0460
		C/S 2 Supervisors and Technicians										
		Supervision of Mail Processing										
4	0004	Mail Processing	2.1.1	C/S 2 Outputs to CRA Model	USPS-T-9: B-2	Total	BOTH	**DA**	**0035**		BOTH	0820
5	0677	Central Mail Mark-up	2.1.2	C/S 2 Outputs to CRA Model	USPS-T-9: B-2	Rate	BOTH				BOTH	0820
6	0820	Supervision of Mail Processing	2.1	Calculated in Model							BOTH	0284
7	0007	Window Service	2.2	C/S 2 Outputs to CRA Model	USPS-T-9: B-2	Total	BOTH	**DA**	**0040**		BOTH	0284
8	0009	Supervision of Admin. and Support Activities ["A" Report]	2.3	C/S 2 Outputs to CRA Model	USPS-T-9: B-2	Total	BOTH	**DA**	**0525**		BOTH	0284
9	0428	Supervision of Admin. and Support Activities [PESSA]	2.3	Calculated in Model								
10	0483	Supervision of Admin. and Support Activities ["C" Report]	2.3	Calculated in Model								
		Supervision of Collection and Delivery										
		Supervision of City Delivery Carriers										
11	0013	Office	2.4.1.1	C/S 2 Outputs to CRA Model	USPS-T-9: B-2	Total	BOTH	**DA**	**0606**		BOTH	0020
12	0014	Delivery Activities	2.4.1.2	C/S 2 Outputs to CRA Model	USPS-T-9: B-2	Total	BOTH	**DA**	**0046**		BOTH	0020
13	0017	Street Other	2.4.1.3	C/S 2 Outputs to CRA Model	USPS-T-9: B-2	Total	BOTH	**DA**	**0280**		BOTH	0020
14	0018	Network Travel	2.4.1.4	C/S 2 Outputs to CRA Model	USPS-T-9: B-2	Total	BOTH	**DA**	**0054**		BOTH	0020
15	0015	Street Other Load (PRC)	2.4.1.5	C/S 2 Outputs to CRA Model		Total	PRC	**DA**	**0692**		PRC	0020
16	0016	Street Access (PRC)	2.4.1.6	C/S 2 Outputs to CRA Model		Total	PRC	**DA**	**0048**		PRC	0020
17	0680	Street SCS Access (PRC)	2.4.1.7	C/S 2 Outputs to CRA Model		Total	PRC	**DA**	**0683**		PRC	0020
18	0020	Supervision of City Delivery Carriers	2.4.1	Calculated in Model							BOTH	0821
19	0674	Rural Delivery Carriers	2.4.2	C/S 2 Outputs to CRA Model	USPS-T-9: B-2	Total	BOTH	**DA**	**0072**		BOTH	0821
20	0675	Vehicle Service	2.4.3	C/S 2 Outputs to CRA Model	USPS-T-9: B-2	Total	BOTH	**DA**	**0057**		BOTH	0821
21	0821	Supervision of Collection and Delivery	2.4	Calculated in Model							BOTH	0284

TABLE II. DISTRIBUTION KEY MASTER LIST EXTRACT

Row Seq.	Comp. Number	Distribution Key Title	Distribution Key Description	Component Affected	Addends	Used in "A" Module	Used in "F" Module	Used in "B" Module	Used in Roll-forward
1	0975	***Computer Forwarding Systems***		1231, 1261, 1201		X	X		
2	0976	***OCR***		1233, 1262, 1202		X	X		
3	0977	***Mail Processing BCS***		1234, 1264, 1204		X	X		
4	0978	***Delivery BCS***		1236, 1266, 1206		X	X		
5	0979	***Carrier Sequence BCS***		1235, 1265, 1205		X	X		
6	0980	***LSM***		1237, 1267, 1207		X	X		
7	0981	***FSM***		1238, 1268, 1208		X	X		
8	0982	***Parcel Sorting & NMO Machines***		1239, 1269, 1209		X	X		
9	0983	***Facer/ Canceler Letters***		1240, 1270, 1210		X	X		
10	0984	***Facer/ Canceler Flats***		1241, 1271, 1211		X	X		
11	0985	***Culling***		1242, 1272, 1212		X	X		
12	0986	***Sack Sorting Machine***		1243, 1273, 1213		X	X		
13	0987	***Small Parcel & Bundle Sorter***		1244, 1274, 1214		X	X		
14	0988	***Remote Barcode System***		1245, 1275, 1215		X	X		
15	0989	***Air Dispatch***		1246, 1276, 1216		X	X		
16	0990	***Strapping***		1247, 1277, 1217		X	X		
17	0991	***Powered Transport Equipment***		1252, 1282, 1222		X	X		
18	0992	***BMC General & Logistics***		1248, 1278, 1218		X	X		
19	0993	***Non BMC General & Logistics***		1249, 1279, 1219		X	X		
20	0035	***Mail Processing***		1250, 1280, 0004, 0031, 1069, 1169		X	X		
21	1259	***Total Maintenance Labor Distribution Key***	Sum of Distributed Equipment Maintenance Labor	0075	X	X		X	
22	1289	***Total Parts & Supplies Distribution Key***	Sum of Distributed Parts and Supplies Keys	0184	X	X			
23	0466	***C/S 3 Scheme Training Distribution Key***	Sum Mail Proc Shape Keys	0467	X	X			
24	0996	***Mail Processing - Letters/Flats***		0468		X			
25	0997	***Mail Processing - Parcels***		0469		X			
26	0040	***Window Services***		0007, 1054, 1154		X	X		
27	0043	***In-Office Direct Labor***		0044, 0049		X			
28	0046	***Delivery Activities***		0014, 0083, 0092, 0101, 0128, 0137, 0222, 0050 (PRC Only)		X		X	

TABLE III. DISTRIBUTION KEY ADDENDS LIST EXTRACT

			H.L. Supv & Admin Clerks General Office-Dist. Key	Quality Control/Revenue Protection Distribution Key	PESSA Dist Key - Servicewide Benefits	All Labor - Distribution Key for E&LR, T&A (2), T&A (3)	Servicewide Benefits - Distribution Key	PESSA Dist Key - for E&LR, T&A (2), T&A (3)
		Distribution Key Number >>>>>>>>	***0294***	***0295***	***0433***	***0525***	***0526***	***0527***
		Number of Addends >>>>>>>>>>>>>	***58***	***3***	***78***	***67***	***78***	***67***
Row Seq.	***Comp. Number***	***Component Title***						
1		***C/S 1 Postmasters***						
2	0001	Postmasters EAS 23 & Below			X	X	X	X
3	0002	Postmasters EAS 24 & Above			X	X	X	X
4	0251	***C/S 1 Postmasters***						
5		***C/S 2 Supervisors and Technicians***						
6	0004	Mail Processing	X		X	X	X	X
7	0677	Central Mail Mark-up	X		X	X	X	X
8	0007	Window Service	X		X	X	X	X
9	0009	Supervision of Admin. and Support Activities ["A" Report]					X	
10	0427	Supervision of Admin. and Support Activities [PESSA]						
11	0483	Supervision of Admin. and Support Activities ["C" Report]			X			
12		Supervision of Collection and Delivery						
13		Supervision of City Delivery Carriers						
14	0013	Office	X		X	X	X	X
15	0014	Delivery Activities	X		X	X	X	X
16	0017	Street Other	X		X	X	X	X
17	0018	Network Travel	X		X	X	X	X
18	0015	Street Other Load (PRC)	X		X	X	X	X
19	0016	Street Access (PRC)	X		X	X	X	X
20	0680	Street SCS Access (PRC)	X		X	X	X	X
21	0674	Rural Delivery Carriers	X		X	X	X	X
22	0675	Vehicle Service	X		X	X	X	X
23		Tech. Personnel & Other Sup. Activities						
24	0029	Employee & Labor Relations ["A" Report]					X	
25	0427	Employee & Labor Relations [PESSA]						
26	0528	Employee & Labor Relations ["C" Report]			X			
27	0030	Higher Level Supervisors			X	X	X	X
28	0031	General Supervision of Mail Processing	X		X	X	X	X
29	0032	General Supervision of Collection & Delivery	X		X	X	X	X
30	0601	Supervisor Training			X	X	X	X
31	0676	Quality Control/Revenue Protection	X		X	X	X	X
32	0678	Joint Supervision Clerks & Carriers	X		X	X	X	X
33	0033	Product Specific and Other S & T	X		X	X	X	X

TABLE IV. D REPORT PROCESSING TABLE

Com-ponent Number	*Component Title*	*Sub-Segment Number*	*"D" Module Routines*	"D" DK or Source Component	"D" Add'l Parms (Part 1)	"D" Add'l Parms (Part 2)	Printed in "D" Report	RF Only
0460	**Total Volume Variable ["A" & "C" Report]**	21					BOTH	
0501	Final Adjustments	21	AA	0460	1506	0199	BOTH	
0506	Final Adjustments [Rollforward Use]	21						BOTH
0504	Adjusted Volume Variable Costs	21	AD	0460	0506	0501	BOTH	
0503	Contingency	21	AA	0504	0576		BOTH	
0502	Total Including Contingency	21	AD	0504	0503		BOTH	
0493	Total Product Specific	21	PS	0199				
0494	Contingency [Product Specific]	21	AA	0493	0576			
0657	Product Specific Including Contingency	21	AD	0493	0494		PRC	
0642	Volume Variable Total Including Contingency	21	AD	0502	0657		PRC	

Table V. CRA Routines Used in the A, F, and B Modules

DESCRIPTION OF CRA MODEL ROUTINES

Phase 1 Process the following before proceeding with subsequent phases.

AV Use the variability factor on the Comp Master to develop non-variable portion (class 199) of a distribution key and recalculate the total
i.e. (1 - factor) / factor * total variable (class 198) = non-variable portion (class 199)

Phase 2 Process the following after all **Phase 1** routines are completed (Order is determined by DK Master worksheet):

DA Distribute the total in the subject component based on amounts in all non-total classes of the distribution key component. This distributes all costs, volume variable and institutional, in the subject component.

DV Distribute costs in the subject component based on amounts in all non-total classes of the distribution key component, excluding institutional cost class 199
This distributes only the volume variable costs in the subject component.

DZ Distribute costs in the subject component based on amounts in all non-total classes of the distribution key component. with the amount to be fully distributed determined by another component

1. Calculate the non-variable portion based on the given factor component that contains the proportion of costs to set aside as insitutional cost.
2. Distribute all remaining costs to all mail classes, including institutional cost class 199, based on the given Distribution Key component
3. Add the non-variable portion back to insitutional cost class 199

DX Distribute costs in the subject component based on amounts in all non-total classes of the distribution key (excluding instituitonal cost class 199) with variability determined by another component

1. Calculate the non-variable portion based on the given factor component that contains the variability for the subject component.
2. Distribute the remaining volume variable costs based on the given Distribution Key
3. Add the non-variable portion back to class institutional cost class 199

CV Compute the non-variable portion (institutional cost class 199) of a component based on a second component
i.e. Total Variable (class 198) / Total (class 200)

Phase 3 Process the following after all **Phase 2** routines are completed:

CP Calculate PESSA Costs.

1. Subtract the variable classes of one component (Pre Pessa) from the comparable classses of a seconf component and store results in the recipient component.
2. Set the non-variable portion of the Pre PESSA component equal to the original total minus the sum of volume variable classes in the recipient component.

Phase 4 Process the following (in any order) after all) **Phase 3** routines are completed:

ST Set component total (class 200) equal to the total (class 200) of another component.

SA Set all values in a component equal to the values in another component.

AJ Multiply the Subject Component by Factor Component, Add results to subject component and subtract results from a specified adjustment class

Table VI. CRA Routines Used in D Report Processing

CRA MODEL ROUTINES USED IN THE D REPORT

AA Multiply a component by a factor component and place results in the subject component
Subtract results from the subject class if the third parameter contains a class.

AD Add two or three components together and place results in the subject component

PS Calculate total Product Specific costs for the PRC CRA.

1. Set institutional cost class 199 in the subject component to 0.
2. Sum up all attributable cost classes in the subject component and put that in institutional cost class 199
3. Multiply all attributable cost classes in subject component by -1 to reverse sign.

III. VBA PROGRAM OVERVIEW

This section gives an overview of the CRA.Model.xls program, with a short description of the primary subroutines used in each main section of the program.

A. Main Program (ModDriver)

Subroutine Driver in module ModDriver is the main program loop for the CRA model. It calls the initialization routines, and loops through the modules, calling the applicable calculation subroutine to implement each module.

B. Initialization Routines (ModInit)

- Auto_Open – Sets up public variables containing the most basic information needed by other routines.

- Init - performs initializations used through program, reads the values of variables that are used to set up internal arrays, then resets the sizes of those arrays, reads the layout of the Data Input Module, reads data from the Control Table, initializes the Components and Mail Class arrays, and calls the Data Checking routine.

- ReadLoaderModelData – reads the name and path of the currently-selected model from the Data Input Module and copies the entire list of available model names and locations to the CRA workbook.

- CheckPhase1 – verifies that every named range that the program is going to have to reference exists. Any error encountered during this checking is a fatal error.

- CheckPhase2 – verifies that all the workbooks and worksheets within each workbook that will be referenced during the run are present.

- ReadControlTable – controls the reading of information from the Control File, including all component and distribution key data.

C. Calculations (ModRoutines)

- RoutineDriver – controls calling the routines for all the phases of the computations
- Phase1 – calls the Phase 1 routines (AV)
- Phase2 – calls the Phase 2 routines (DA, DV, DZ, DX, and CV).
- Phase3 – calls the Phase 3 routines (CP)
- Phase4 – calls the Phase 4 routines (ST, SA, and AJ)
- RunDRoutines – calls the routines necessary for the "D" report (AA, AD, and PS

D. SubTotals (ModSubtotals)

- ClassSubtotals – develops cell formulas that calculate Mail Class subtotals.
- ComponentSubtotals – develops cell formulas that calculate subtotals for components.

E. Reporting (ModReports)

- RptInit – initializes variables in preparation for developing reports
- Reports – formats the reports. This subroutine takes data from the OutputMatrix of the applicable workbook and formats it for printed output. No calculations are done for the reporting process.

F. Writing Output Workbooks (ModWorkbooks)

- MakeWorkingBook – opens or creates the workbook of interest, as indicated by the module being executed.
- OpenWB – opens the workbook specified in the arguments. If the workbook does not exist, this subroutine creates it.
- MakeInputMatrixSheet – creates the InputMatrix worksheet, which contains the inputs for this module.
- AddCSSheets – adds the worksheet for each Cost Segment to the current workbook.

- MakeOutputMatrixSheet – creates the OutputMatrix worksheet, which will be used to generate reports.

- FillDKSheet – similar to AddCSSheets, but is specific to the Distribution Key worksheets.

- UpdateOutputMatrix – updates sheet OutputMatrix for calculations performed in the current workbook.

G. Response to User Actions (ModUserInterface)

- UserSelectModel – adjusts appropriate variables when the user selects or changes the selection of a model to use

- ChangeRunType – adjusts appropriate variables when the user selects or changes the selection of Run Type.

- Run – controls overall program flow (including the calling of the Driver routine) when the user clicks the "Run" button.

- RunCleanup – performs housekeeping tasks at the end of a run.

- GetRFOptions – prompts for options when the user chooses to perform a Rollforward run.

H. Utility Functions (ModUtil)

- IsClass – determines whether the argument is a valid class

- IsComponent – determines whether the argument is a valid component

- SaveCloseWorkbooks – saves and closes the workbooks that the program had to open or create during the run (if the user selects the Save/Close option)

I. Error Handling Routines (modErrorHandling)

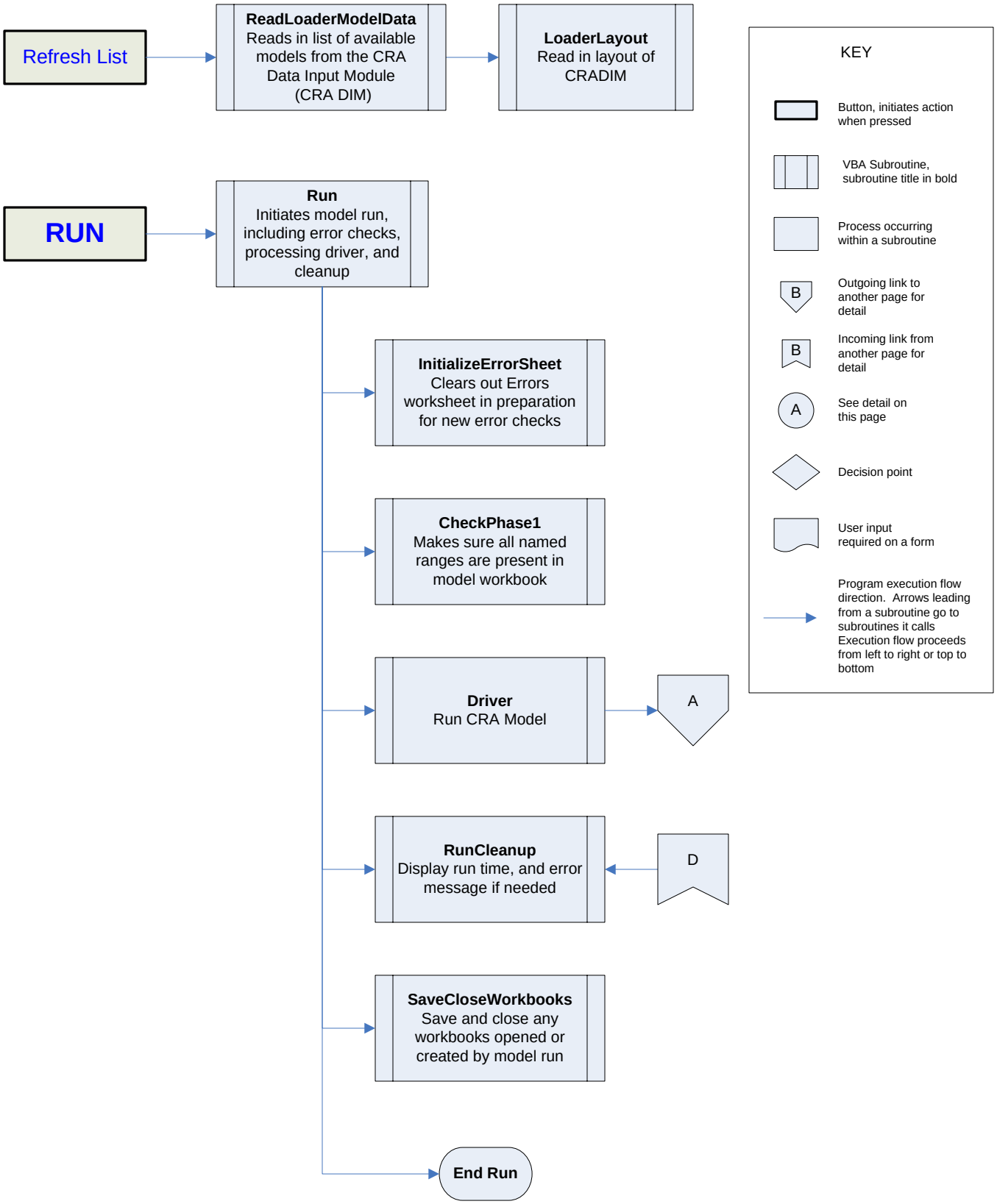
- InitializeErrorSheet – initializes the Errors worksheet of the CRA workbook by clearing all previous entries and recording the starting date/time of the run.

- RecordError – writes an error onto the Errors page of the CRA workbook.

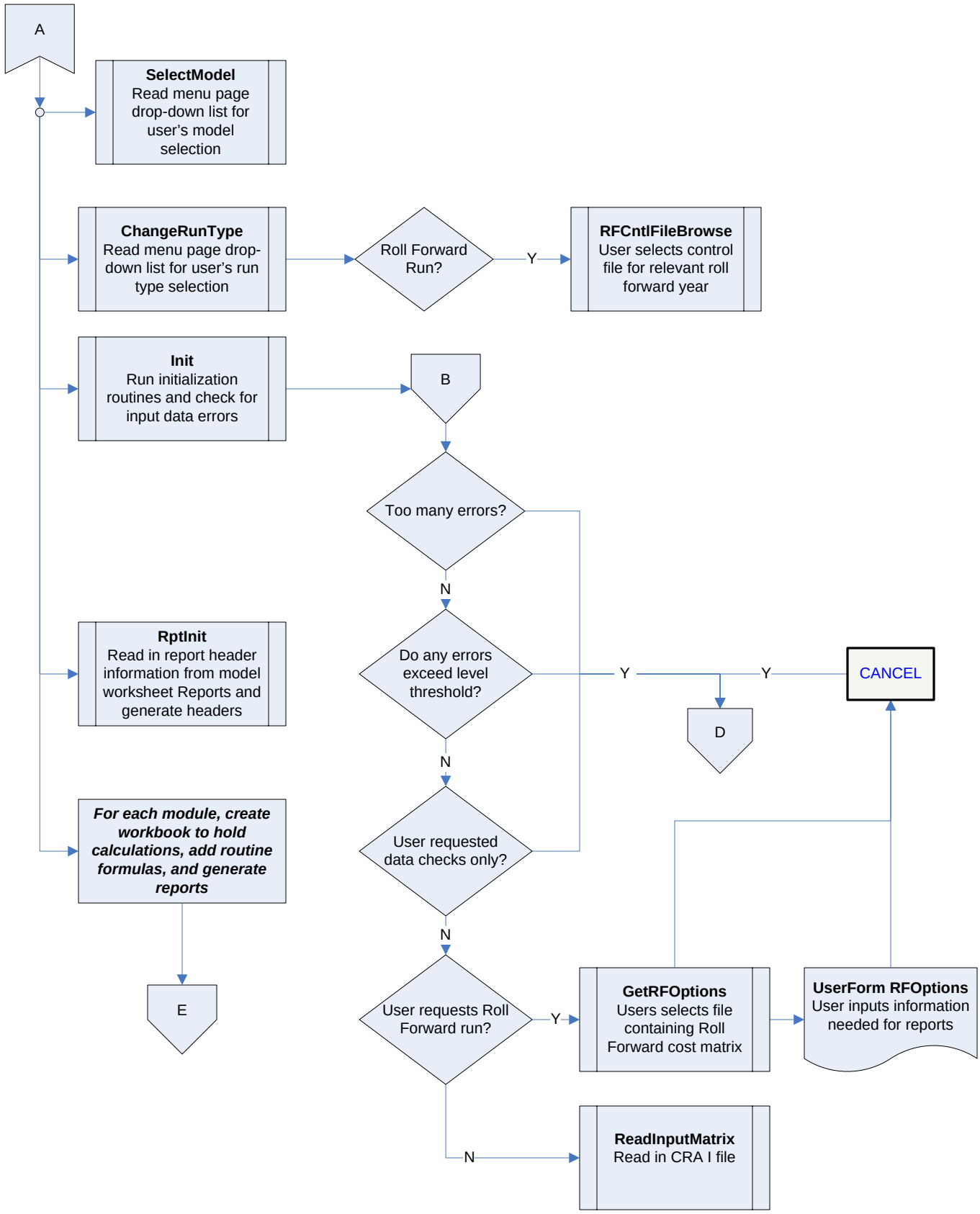
- DataCheck – performs data logic and consistency checks on data that has been read from the Control workbook.

- CheckRoutine – checks to see if the arguments for the specified routine are valid.

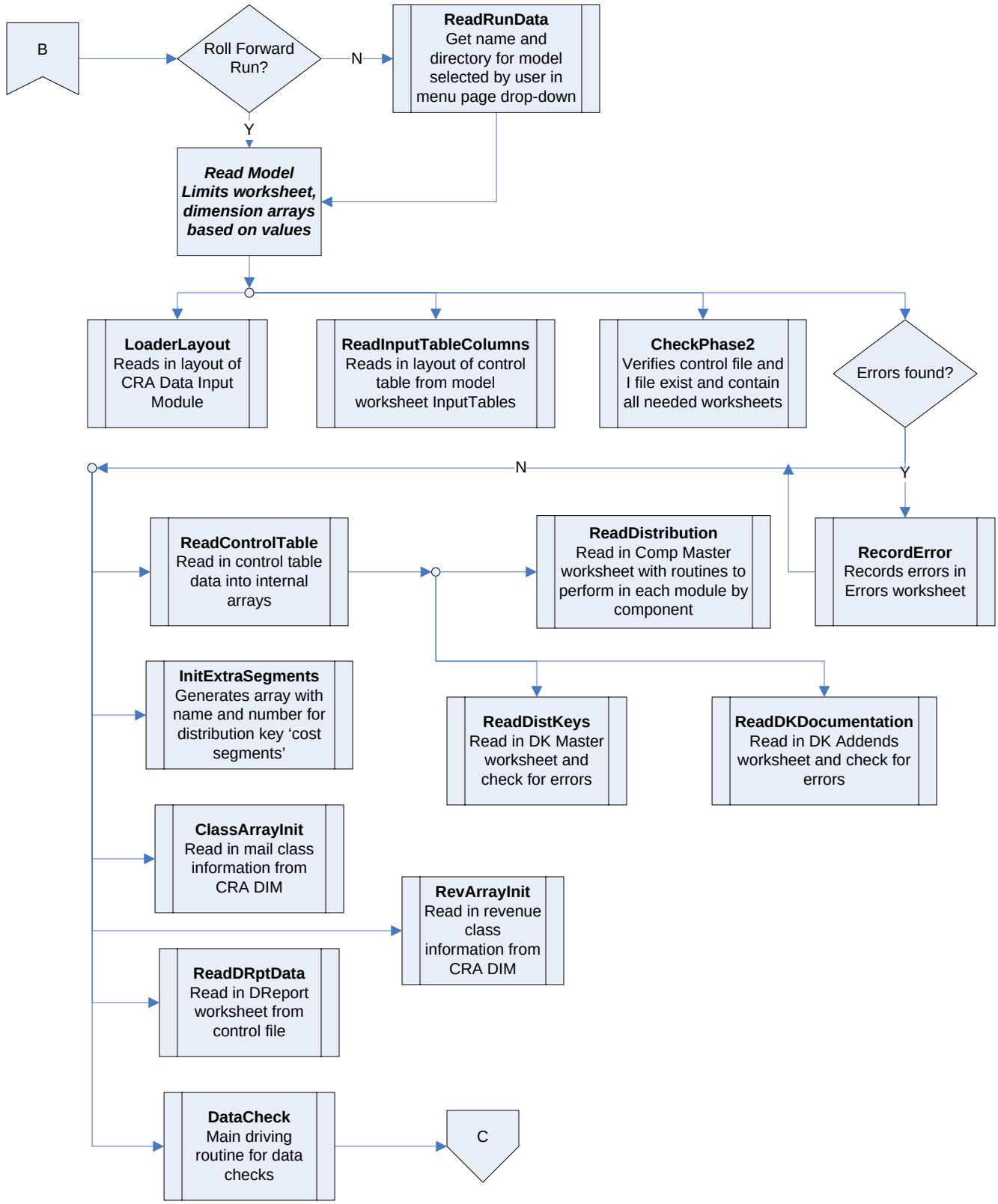
IV. CRA MODEL FLOW DIAGRAM



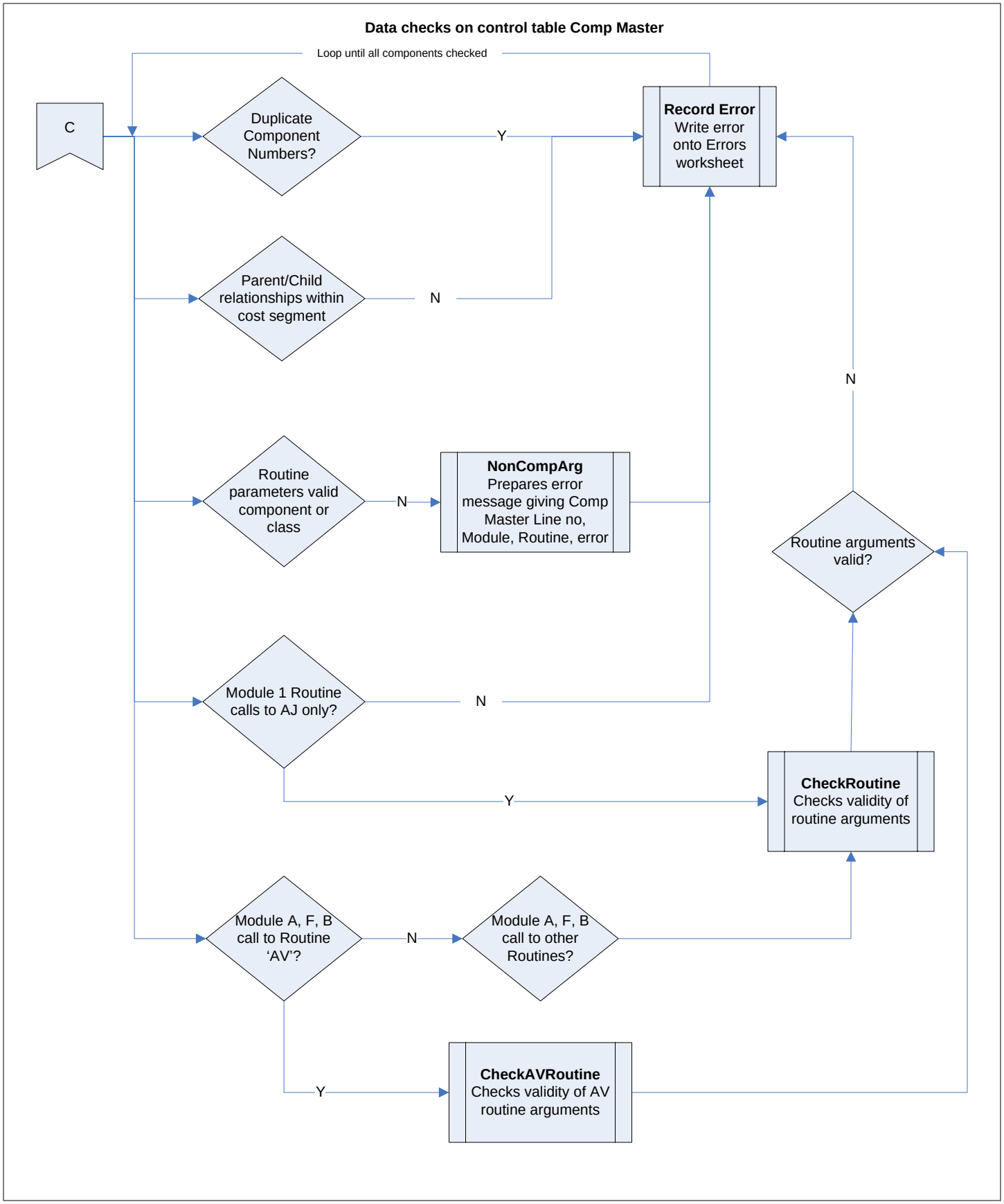
CRA MODEL FLOW DIAGRAM
Driver Detail



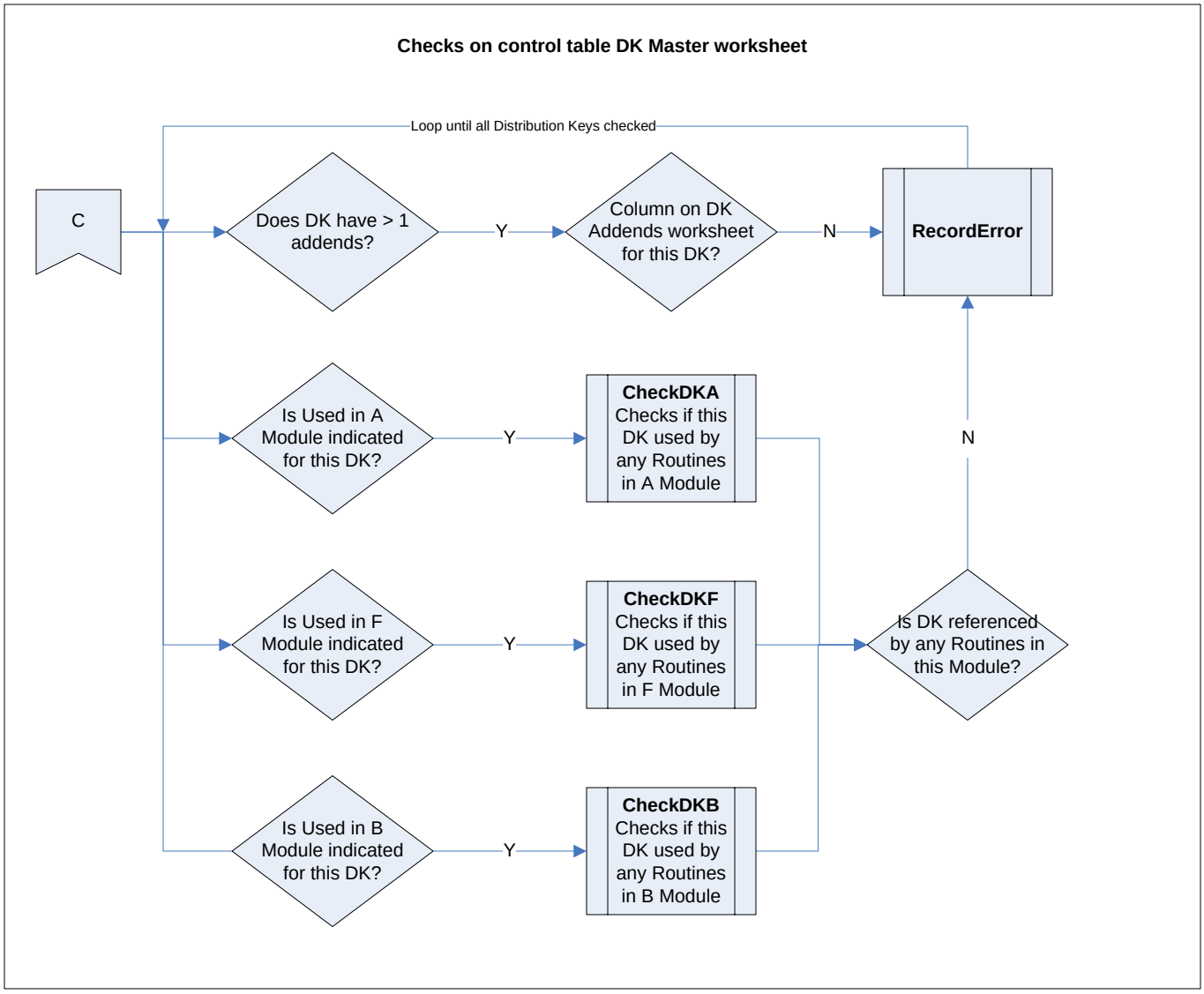
CRA MODEL FLOW DIAGRAM
Initializations Detail



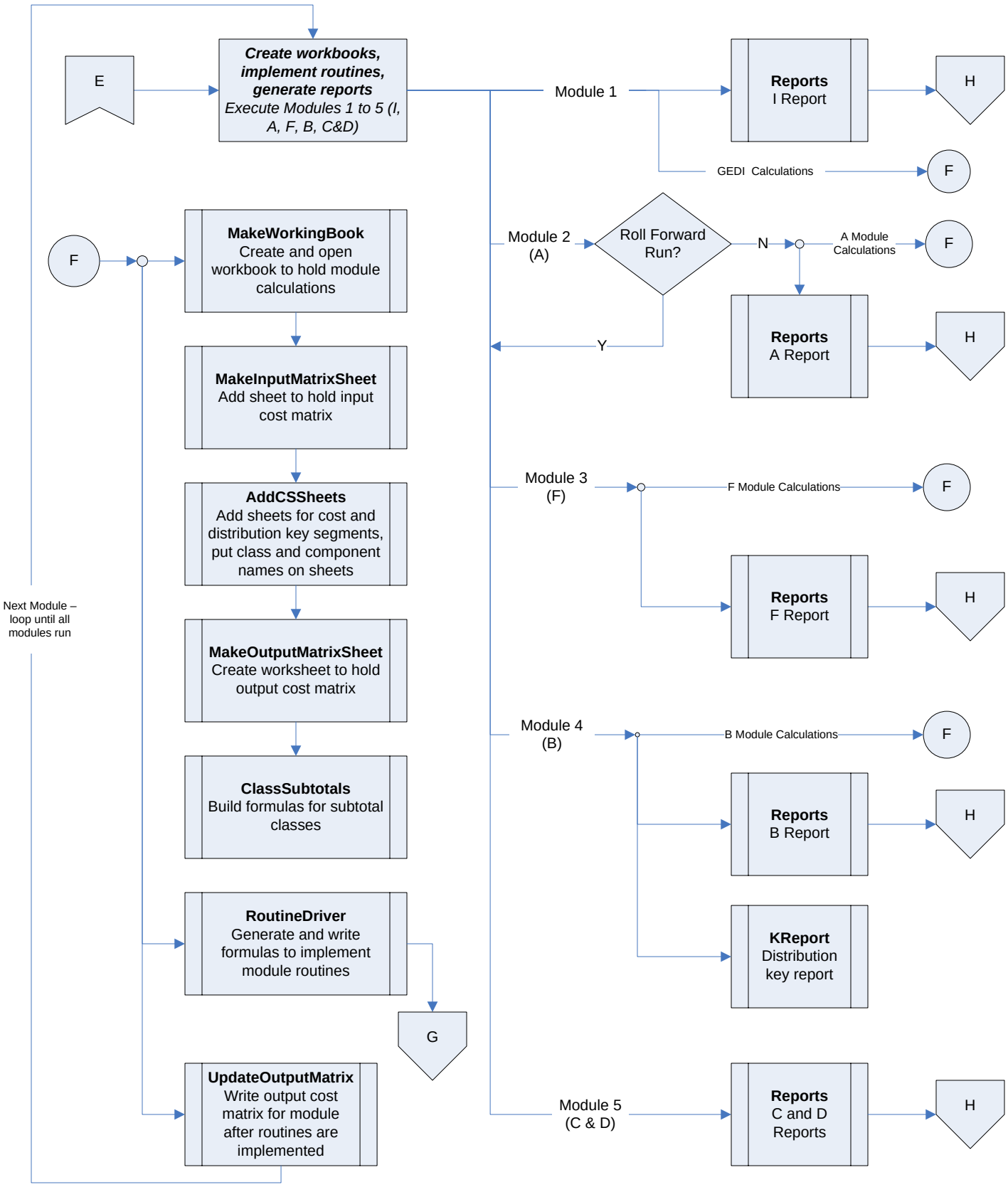
CRA MODEL FLOW DIAGRAM
Data Check Detail – Comp Master



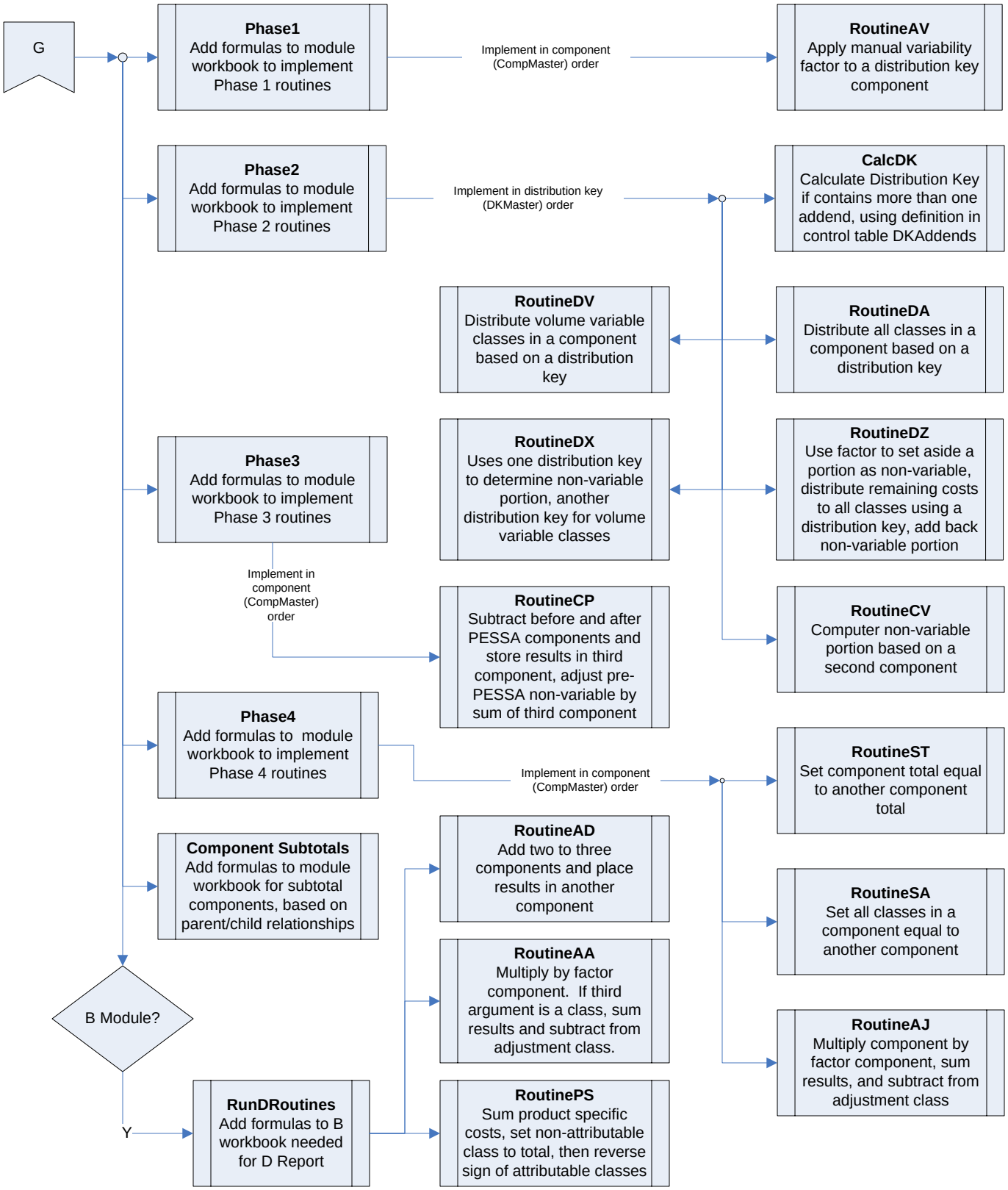
CRA MODEL FLOW DIAGRAM
Data Check Detail – DK Master



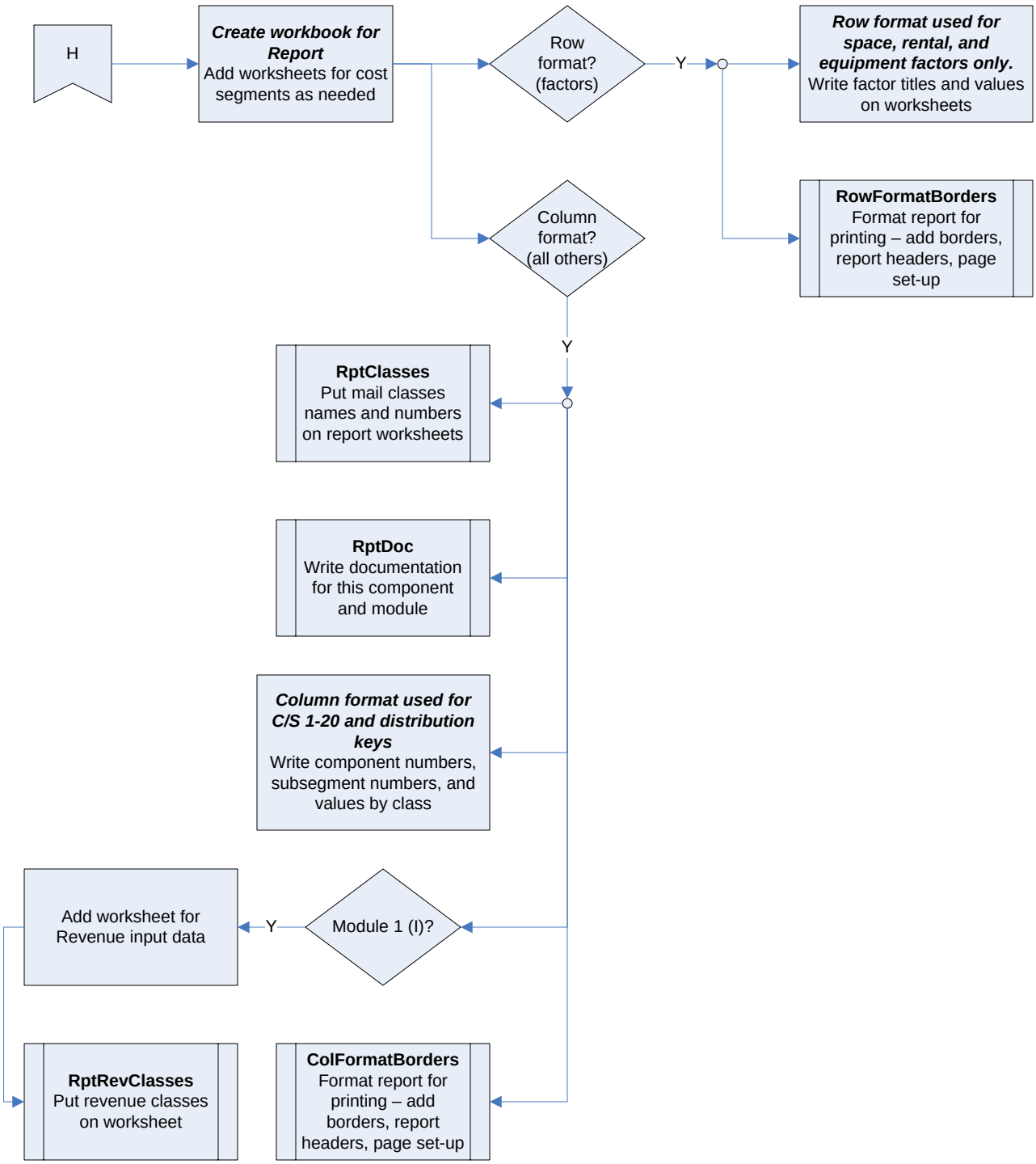
CRA MODEL FLOW DIAGRAM
CRA Calculations and Reports Detail



CRA MODEL FLOW DIAGRAM
CRA Calculations and Reports – Routine Driver Detail



CRA MODEL FLOW DIAGRAM
CRA Calculations and Reports – Reports Detail



V. VBA PROGRAM DESCRIPTION

This section describes each Visual Basic for Applications subroutine in CRA.Model.xls. The code for the subroutines is contained in program modules. A description of the purpose for each subroutine is shown, as well as the workbooks that are used as input, the global variables used as input, and if relevant, a description of the variables passed to this subroutine by another subroutine. Each section also lists the subroutines called by this subroutine, as well as the subroutines that call this subroutine. Finally, each section shows the workbooks that are changed or output by the subroutine and any global variables that are initialized.

A. Main Program - ModDriver

1. Subroutine: Driver()

   Purpose:

   - Calls initialization routines.
   - Runs the main program loop, which creates output workbooks for each applicable module.
   - Calls the Reports subroutine, which reformats output for printing.

   Inputs:

   Public Variables:

   ErrDat.Level
   RunOK
   iRunType

   Outputs:

   Public Variables:

   ErrStop
   RunOK
   iModule

   Subroutines Called:

   Auto_Open
   SelectModel
   ChangeRunType
   Init
   GetRFOptions
   ReadInputMatrix
   RptInit
   MakeWorkingBook
   RoutineDriver
   UpdateOutputMatrix

Reports
KReports
DReports

Called By:

Run

B. Initialization Routines (ModInit)

1. Subroutine: Auto_Open()

   Purpose:

   - Preliminary initization

   Inputs:

   Public Variables:

   CRAPath
   ErrDat.Level

   Outputs:

   Public Variables:

   ErrStop
   ErrDat.Level

   Subroutines Called:

   SetRowColVariables

   Called By:

   Excel (when workbook is opened)
   Driver

2. Subroutine: ReadLoaderModelData()

   Purpose:

   - Reads the name and path of the currently-selected model from the Data Entry Model.
   - Copies the entire list of available model names and paths from the Data Entry Model.

   Inputs:

   Public Variables:

   CRABook
   ErrDat.Level

   Workbooks:

   Control workbook

   Outputs:

   Public Variables:

   CRAModels

ModName
ModPath

Subroutines Called:

OpenWB

Called By:

Event (User presses Refresh Lists button on menu page)

3. Subroutine: Init()

Purpose:

- Initization

Inputs:

Public Variables:

iRunType

Outputs:

Public Variables:

SaveCloseNames
SaveCloseFlag
ErrMaxAllow
MaxClass
MaxCS
MaxCompon
MaxDKComp
MaxComps
MaxDefs
TotVVClass
OtherClass
TotClass
SkipClass
ComponentData(MaxCompon)
CostMatrix(MaxClass, MaxComps)
CompColumn(MaxComps)
CompSheet(MaxComps)

Subroutines Called:

InitMtxChange
LoaderLayout
ReadInputTableColumns
ReadControlTable
InitExtraSegments
ClassArrayInit
RevArrayInit
ReadDRptData

DataCheck

Called By:

Driver

4. Subroutine: CheckPhase1()

Purpose:

- Preliminary error checking: looks for all the named ranges expected in the CRA workbook.

Inputs:

None

Outputs:

Public Variables:

ErrDad

Subroutines Called:

RecordError

Called By:

Run

5. Subroutine: CheckPhase2()

Purpose:

- Second level error checking: looks to see that ll workbooks needed are present and that all worksheets in those workbooks are present.

Inputs:

Public Variables:

CntlBook
iRunType
ModName
ModPath

Outputs:

Public Variables:

ErrDat

Subroutines Called:

OpenWB
RecordError

Called By:

Init

6. Subroutine: ReadControlTable()

Purpose:

- Controls reading of information from the Control workbook

Inputs:

None

Outputs:

None

Subroutines Called:

ReadDistribution
ReadDistKeys
ReadDKDocumentation

Called By:

Init

7. Subroutine: ReadDistribution()

Purpose:

- Reads data from the Master Components worksheet

Inputs:

Public Variables:

CntlCompBook
CntlCompSheet
xlMaxLine
NumberCol
TitleCol
SubSegmentCol
SourceCol
LibRefCol
InputTypeCol
ManualVariabilityCol
PrintICol
Module1Col

Distrib1Col
Misc1Col
ModuleACol
DistribACol
MiscACol
PrintACol
ParentACol
ModuleFCol
DistribFCol
MiscFCol
PrintFCol
ParentFCol
ModuleBCol
DistribBCol
MiscBCol
PrintBCol
ParentBCol
PrintCCol
ParentCCol

Outputs:

Public Variables:

ComponentData()
CompIndx
CSCount
CostSegment
CSIndex

Subroutines Called:

OpenWB

Called By:

ReadControlTable

8. Subroutine: ReadDistKeys()

Purpose:

- Reads data from the DK Master sheet of the Control Workbook

Inputs:

Public Variables:

CntlBook
CntlDKSheet
xlMaxLine
DKCompCol
DKTitleCol
DKContentsCol
DKAffectedCol
DKAddendsCol
DKACol
DKFCol
DKBCol
DKRFCol
DKMax

Outputs:

Public Variables:

DKData()
DKIndx()
ErrDat

Subroutines Called:

RecordError

Called By:

ReadControlTable

9. Subroutine: ReadDKDocumentation()

Purpose:

- Reads data from the DK Addends worksheet in the Control workbook

Inputs:

Public Variables:

CntlBook
CntlDKDetailSheet

xlMaxLine
DKACompCol
DKADataCol
DKAKeyRow
DKAAddendCountRow
DKAFirstDataRow

Outputs:

Public Variables:

DKDetail
DetailMax
ErrDat

Subroutines Called:

None

Called By:

ReadControlTable

10. Subroutine: InitExtraSegments()

Purpose:

- Initialization of "cost segments" above 20.

Inputs:

Public Variables:

CompsMax
LastCS
ComponentData().CS
ComponentData().SubSegment

Outputs:

Public Variables:

PlaceHolderSheets
PlaceHolderTypes
PlaceHolderCount

Subroutines Called:

None

Called By:

Init

11. Subroutine: ReadRunData()

    Purpose:

    - Read data from the CRA Model Menu sheet

    Inputs:

    User-specified Variables:

    Model Name

    Outputs:

    Public Variables:

    ModPath
    ModName

    Subroutines Called:

    None

    Called By:

    Init

12. Subroutine: ReadInputTableColumns()

    Purpose:

    - Reads data from the Input Table worksheet in the CRA workbook pertaining to the layout of the Control workbook

    Inputs:

    Worksheet variables:

    Input Table worksheet

    Public Variables:

    ModName

    Outputs:

    Public Variables:

    NumberCol
    TitleCol
    SubSegmentCol
    SourceCol
    LibRefCol
    InputTypeCol

ManualVariabilityCol
PrintICol
Module1Col
Distrib1Col
Misc1Col
ModuleACol
DistribACol
MiscACol
PrintACol
ParentACol
ModuleFCol
DistribFCol
MiscFCol
PrintFCol
ParentFCol
ModuleBCol
DistribBCol
MiscBCol
PrintBCol
ParentBCol
PrintCCol
ParentCCol
CompMasterFirstDataRow
CntlBook
CntlCompSheet
CntlDKSheet
CntlDKDetailSheet
CntlDReportSheet
DKCompCol
DKTitleCol
DKContentsCol
DKAffectedCol
DKAddendsCol
DKACol
DKFCol
DKBCol
DKRFCol
DKACompCol
DKATitleCol
DKADataCol
DKAFirstDataRow
DKAKeyRow
DKAAddendCountRow
IFileSheetName
DCompCol
DRoutineCol
DParm1Col
DParm2Col
DParm3Col
DPrintCol
DRFOnlyCol

Subroutines Called:

None

Called By:

Init

13. Subroutine: LoaderLayout()

Purpose:

- Reads data about how the Data Entry Module is set up

Inputs:

Worksheet Variables:

InputTable worksheet

Outputs:

Public Variables:

LoaderName
LoaderModelsSheet
LoaderClassRange
LoaderCurrentModel
LoaderCurrentDir
LoaderModelsRange
LoaderRevRange

Subroutines Called:

None

Called By:

Init

14. Subroutine: ClassArrayInit()

Purpose:

- Initializes the Mail Classes arrays

Inputs:

Workbooks:

Data Entry Module

Outputs:

Public Variables:

MailClassData
ClassIndx

ClassMax
ClassRow

Subroutines Called:

OpenWB

Called By:

Init

15. Subroutine: ReadInputMatrix()

Purpose:

- Reads data from the IMatrix

Inputs:

Workbook:

Input (I-Matrix) workbook

Public Variables:

ModName
ModPath
MatrixRowOffset
MatrixColOffset

Outputs:

Public Variables:

CostMatrix

Subroutines Called:

None

Called By:

Driver

16. Subroutine: SetRowColVariables()

Purpose:

- Sets the value of row and column headers and other constants

Inputs:

None

Outputs:

Public Variables:

HeadRow
CompNameRow
CompNoRow
DocRow
VariabilityRow
ClassLabelRow
FirstRow
SeqCol
ClassNameCol
ClassNoCol
ColHeadCol
FirstCol
MatrixRowOffset
MatrixColOffset
ModuleLabel()

Subroutines Called:

None

Called By:

Auto_Open

17. Subroutine: InitMtxChange()

Purpose:

- Sets the entire MtxChange array to False

Inputs:

Public Variables:

MaxComps

Outputs:

Public Variables:

MtxChange

Subroutines Called:

None

Called By:

Init
MakeOutputMatrixSheet

18. Subroutine: ReadRptData()

    Purpose:

    - Initializes the DData array

    Inputs:

    Workbook:

    Control workbook

    Public Variables:

    CntlBook
    CntlDReportSheet
    HeadRow
    DCompCol

    Outputs:

    Public Variables:

    DData
    DMax

    Subroutines Called:

    None

    Called By:

    Run

19. Subroutine: RevArrayInit()

    Purpose:

    - Initializes the Revenue Classes array

    Inputs:

    Workbooks:

    Data Entry Module

    Public Variables:

    LoaderName
    CRAPath
    LoaderRevRange

    Outputs:

Public Variables:

RevClassData
RevMax

Subroutines Called:

OpenWB

Called By:

Init

C. Calculations (ModRoutines)

1. Subroutine: RoutineDriver()

   Purpose:

   - Main driving subroutine for writing formulas to the CRA workbooks.
   - Calls the four Phase routines

   Inputs:

   Public Variables:

   iModule

   Outputs:

   None

   Subroutines Called:

   Phase1
   Phase2
   Phase3
   Phase4
   RunDRoutines

   Called By:

   Driver

2. Subroutine: Phase1()

   Purpose:

   - Calls the Phase 1 routine (AV) for each Module

   Inputs:

   Public Variables:

   CompsMax
   IModule
   ComponentData

   Outputs:

   Public Variables:

   MtxChange

   Subroutines Called:

   None

Called By:

RoutineDriver

3. Subroutine: Phase2()

Purpose:

- Calls the Phase 2 routines (DA, DV, DZ, DX, and CV) for each Module

Inputs:

Public Variables:

iDK
MaxDefs
iModule
DKData
ComponentData

Outputs:

Public Variables:

MtxChange

Subroutines Called:

CalcDK
RoutineDA
RoutineDV
RoutineDZ
RoutineDX
RoutineCV

Called By:

RoutineDriver

4. Subroutine: Phase3()

Purpose:

- Calls the Phase 3 routines (CP)

Inputs:

Public Variables:

CompsMax
ComponentData
iModule

Outputs:

Public Variables:

MtxChange

Subroutines Called:

RoutineCP

Called By:

RoutineDriver

5. Subroutine: Phase4()

Purpose:

- Calls the Phase 4 routines (ST, SA, and AJ)

Inputs:

Public Variables:

CompsMax
ComponentData
iModule

Outputs:

Public Variables:

MtxChange

Subroutines Called:

RoutineST
RoutineSA
RoutineAJ

Called By:

RoutineDriver

6. Subroutine: RoutineAV(SC, VF)

Purpose:

- Use Varibility Factor (VF) to calculate non-variable portion of a component (SC)

Inputs:

Public Variables:

CompSheet
CompColumn
ClassMax
MailClassData

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Phase1

7. Subroutine: RoutineDA(SC, DK)

Purpose:

- Distribute Total in subject component (SC) based on all non-total classes of distribution key (DK)

Inputs:

Public Variables:

CompColumn
CompSheet
MailClassData
ClassMax
ClassRow
TotClass

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Phase2

8. Subroutine: RoutineDV(SC, DK)

Purpose:

- Distribute Total in subject component (SC) based on all non-total classes of distribution key (DK) excluding class 199.

Inputs:

Public Variables:

CompColumn
CompSheet
MailClassData
ClassMax
ClassRow
TotClass
TotVVClass
TotVVRow

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Phase2

9. Subroutine: RoutineDZ(SC, DK1, DK2)

Purpose:

- Distribute Total in subject component (SC) based on all non-total classes of distribution key (DK1) including class 199 with amount to be fully distributed determined by another component (DK2)

Inputs:

Public Variables:

CompColumn
CompSheet
MailClassData
ClassMax
ClassRow
TotClass
TotRow
TotVVClass

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Phase2

10. Subroutine: RoutineDX(SC, DK2, DK2)

Purpose:

- Distribute Total in subject component (SC) based on amounts in all non-total classes of the distribution key (DK1) excluding class 199 with variability determined by another component (DK2)

Inputs:

Public Variables:

CompColumn
CompSheet
MailClassData
ClassMax
ClassRow
TotVVRow
TotVVClass

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Phase2

11. Subroutine: RoutineCV(SC, DK)

Purpose:

- Compute non-variable portion (class 199) of a component (SC) based on a second component (DK)

Inputs:

Public Variables:

CompColumn
CompSheet
MailClassData
ClassMax
ClassRow
TotVVClass

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Phase2

12. Subroutine: RoutineCP(SC, PC, PP)

Purpose:

- Calculate PESSA Costs.
- Subtract the variable classes of pre-PESSA component (PP) from the comparable classses of the after-PESSA component (PC) and store results in the subject component (SC). Set the non-variable portion of the pre- PESSA component equal to the original total minus the sum of variable classes in the subject component.

Inputs:

Public Variables:

CompColumn
CompSheet
MailClassData
ClassMax
ClassRow
TotClass
TotVVClass

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Phase3

13. Subroutine: RoutineST(SC, AC)

Purpose:

- Set component (SC) total (class 200) equal to the total (class 200) of another component (AC)

Inputs:

Public Variables:

CompColumn
CompSheet
ClassRow
TotClass

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Phase4

14. Subroutine: RoutineSA(SC, AC)

Purpose:

- Set all values in a component (SC) equal to the values in another component (AC)

Inputs:

Public Variables:

CompColumn
CompSheet
MailClassData
ClassMax
ClassRow

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Phase4

15. Subroutine: RoutineAJ(SC, FC, AC)

Purpose:

- Multiply subject component (SC) by factor component (FC), add results to subject component and subtract the results from the adjustment class (AC)

Inputs:

Public Variables:

CompColumn
CompSheet
CompNameRow
ClassMax
MailClassData

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Phase4

16. Subroutine: CalcDK(DK)

Purpose:

- Calculates the distribution key if the number of addends in the key is greater than 1

Inputs:

Public Variables:

CompColumn
CompSheet
DetailMax

DKDetail
MaxAddends
ClassMax
MailClassData
ClassRow

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Phase2

17. Subroutine: RunDRoutines()

Purpose:

- Controls calling routines for the D report

Inputs:

Public Variables:

DMax
DData

Outputs:

Public Variables:

MtxChange

Subroutines Called:

RoutineAA
RoutineAD
RoutinePS

Called By:

RoutineDriver

18. Subroutine: RoutineAA(NC, SC, FC, AC)

Purpose:

- Multiply subject component (SC) by factor component (FC), add results to subject component and subtract results from adjustment class (AC). Place results in new component (NC)

Inputs:

Public Variables:

CompColumn
CompSheet
ClassMax
MailClassData
ClassRow

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

RunDRoutines

19. Subroutine: RoutineAD(SC, A1, A2, A3)

Purpose:

- Add components A1, A2, and A3 and place results in subject component (SC)

Inputs:

Public Variables:

CompColumn
CompSheet
ClassMax
MailClassData
ClassRow

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

RunDRoutines

20. Subroutine: RoutinePS(SC, AC)

Purpose:

- Adjusts subject component (SC) as follows:
  - Set adjustment class (AC) amount to 0
  - Sum up all non-zero classes and put that in AC
  - Reverse signs of all volume variables

Inputs:

Public Variables:

CompColumn
CompSheet
ClassMax
MailClassData
ClassRow

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

RunDRoutines

D. SubTotals (ModSubtotals)

1. Subroutine: ClassSubtotals()

   Purpose:

   - Develops a formula for subtotal classes and inserts the formula into the workbook.

   Inputs:

   Public Variables:

   ClassMax
   MailClassData

   Outputs:

   Public Variables:

   ClassFormulas

   Subroutines Called:

   None

   Called By:

   MakeWorkingBook

2. Subroutine: ComponentSubtotals()

   Purpose:

   - Develops and writes formulas for subtotal components

   Inputs:

   Public Variables:

   CompsMax
   iModule
   ComponentData
   CompColumn
   CompSheet
   ClassRow
   SubTotCol

   Outputs:

   Workbooks:

   Formulas written in appropriate cells of the current workbook

   Public Variables:

MtxChange

Subroutines Called:

None

Called By:

RoutineDriver

E. Reporting (ModReports)

1. Subroutine: RptInit()

Purpose:

- Initializes variables that are used to prepare reports

Inputs:

Public Variables:

CRABook
iRunType

Outputs:

Public Variables:

IHeader1
IHeader2
IStartPrintRows
AHeader1
AHeader2
FHeader1
FHeader2
BHeader1
BHeader2
CHeader1
CHeader2
DHeader1
DHeader2
KHeader1
KHeader2
KHeader3
KComps
LastCS
RptCentury
RptStartRow
RptStartCol
RptCompRow
CalcKeysCS
IPrintDecimals
IRevComp
USPSorPRC
RYear

Subroutines Called:

None

Called By:

Driver

2. Subroutine: Reports()

Purpose:

- Runs the reports (prepares a formatted copy of the workbooks)

Inputs:

Public Variables:

iModule
ComponentData
ModuleLabel
ModName
MaxCompon
IStartPrintRows
RptStartRow
RptStartCol
xlMaxLine
RptCompRow
CostMatrix
iRunType
CSIndex
CompsMax

Outputs:

Workbooks:

Formulas written in appropriate cells of the current workbook

Public Variables:

Subroutines Called:

OpenWB
RptClasses
RptDoc
RptRevClasses
ColFmtBorders
RowFmtBorders

Called By:

Driver

3. Subroutine: RptClasses()

Purpose:

- Inserts the Mail Classes onto the current sheet

Inputs:

Public Variables:

ClassMax
RptStartRow
MailClassData
RptStartCol
iModule

Outputs:

Workbooks:

Mail Class labels written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Reports
KReport
DReport

4. Subroutine: RptRevClasses()

Purpose:

- Inserts the Revenue Classes onto the current sheet

Inputs:

Public Variables:

RevMax
RptStartRow
RptStartCol
RevClassData

Outputs:

Workbooks:

Revenue Class labels written in appropriate cells of the current workbook

Subroutines Called:

None

Called By:

Reports

5. Subroutine: ColFmtBorders(Header1, Header2, Header3)

Purpose:

- Inserts column borders on worksheets where components are printed in columns

Inputs:

Public Variables:

RptStartRow
RptStartCol
RptCompRow
xlMaxLine
RevMax
ClassMax
RevClassData
MailClassData
iModule
KRpt
DRpt

Outputs:

Workbooks:

Column borders inserted in appropriate places in the current report workbook

Subroutines Called:

None

Called By:

Reports
KReport
DReport

6. Subroutine: RowFmtBorders(Header1, Header2, Header3)

Purpose:

- Inserts borders on worksheets where components are printed in rows (rather than columns)

Inputs:

Public Variables:

RptStartCol
RptStartRow
xlMaxLine

Outputs:

Workbooks:

Borders inserted in appropriate places in the current report workbook

Subroutines Called:

None

Called By:

Reports

7. Subroutine: RptDoc(Idx)

Purpose:

- Inserts documentation for the component in the appropriate cell near the top of the column

Inputs:

Public Variables:

iModule
ComponentData

Outputs:

Workbooks:

Component documentation written to appropriate cell in the current workbook

Subroutines Called:

None

Called By:

Reports

8. Subroutine: KReport

Purpose:

- Prepares the "K" report (a special report, not part of the generalized routine)

Inputs:

Public Variables:

ModName
ModPath
KRptComps
RptStartCol
RptStartRow
xlMaxLline

ComponentData
CostMatrix

Outputs:

Workbooks:

Component documentation written to appropriate cell in the current workbook

Subroutines Called:

OpenWB
RptClasses
ColFmtBorders

Called By:

Driver

9. Subroutine: DReport()

Purpose:

- Prepares the "D" report (a special report, not part of the generalized routine)

Inputs:

Public Variables:

ModName
ModPath
DMax
DData
iRunType
USPSorPRC
DRFOnly
RptCompRow
RptStartCol
RptStartRow
ComponentData
CompIndx
ClassRow
CostMatrix

Outputs:

Workbooks:

Component documentation written to appropriate cell in the current workbook

Subroutines Called:

OpenWB
RptClasses

ColFmtBorders

Called By:

Driver

F. Writing Output Workbooks (ModWorkbooks)

1. Subroutine: WriteClasses()

   Purpose:

   - Inserts the Mail Classes onto the current sheet

   Inputs:

   Public Variables:

   FirstRow
   ClassLabelRow
   SeqCol
   ClassNameCol
   ClassNoCol
   MailClassData

   Outputs:

   Workbooks:

   Mail Class Numbers and Names inserted onto the worksheet

   Subroutines Called:

   None

   Called By:

   AddCSSheets

2. Subroutine: MakeOutputMatrixSheet()

   Purpose:

   - Initializes the OutputMatrix sheet in the current workbook
   - Sets the OutputMatrix sheet equal to the InputMatrix sheet
   - Clears the MtxChange array

   Inputs:

   None

   Outputs:

   Workbooks:

   Creates and initializes worksheet OutputMatrix

   Subroutines Called:

   InitMtxChange

Called By:

MakeWorkingBook

3. Subroutine: WriteComps(CS)

Purpose:

- Inserts the component names and numbers onto the current worksheet

Inputs:

Public Variables:

CompsMax
ComponentData
CompNameRow
CompNoRow
DocRow
CompColumn
CompSheet
iModule
ClassMax
MailClassData

Outputs:

Workbooks:

Component Numbers and Names inserted onto the worksheet

Subroutines Called:

None

Called By:

AddCSSheets

4. Subroutine: UpdateOutputMatrix

Purpose:

- Updates sheet OutputMatrix for calculations performed in the current workbook

Inputs:

Public Variables:

MaxComps
MtxChange
ClassMax
MailClassData

CostMatrix
ClassRow
CompColumn

Outputs:

Workbooks:

Updated worksheet OutputMatrix

Subroutines Called:

None

Called By:

Driver

5. Subroutine: OpenWB(OpenBook, OpenPath, Result)

Purpose:

- Creates, opens, or activates the workbook specified in OpenBook and OpenPath
- Returns Result to indicate whether workbook was already open, had to be opened, or had to be created.

Inputs:

Arguments, as described above

Outputs:

Public Variables:

SCCount
SaveCloseNames
SaveCloseFlag
SCFlag

Subroutines Called:

None

Called By:

MakeWorkingBook
Reports
KReport
DReport
GetRFOptions
CheckPhase2
ReadDistribution
ReadInputMatrix
RevArrayInit

6. Subroutine: AddCSSheets()

Purpose:

- Inserts Cost Segment worksheets and places Mail Class labels on each sheet

Inputs:

Public Variables:

ModuleLabel
iModule
CostSegment

Outputs:

Workbooks:

Component Numbers and Names inserted onto the worksheet

Subroutines Called:

WriteClasses
CSSheetCommonHeaders
WriteComps
FillDKSheet

Called By:

MakeWorkingBook

7. Subroutine: CSSheetCommonHeaders()

Purpose:

- Places the common headers on the current worksheet

Inputs:

Public Variables:

CompNameRow
CompNoRow
ColHeadCol
DocRow

Outputs:

Workbooks:

Worksheet headers

Subroutines Called:

None

Called By:

AddCSSheets

8. Subroutine: MakeWorkingBook()

Purpose:

- Driver for opening or creating the working workbook

Inputs:

Public Variables:

ModName
ModuleLabel
iModule
ModPath

Outputs:

Workbooks:

Creates or opens and initializes the working workbook

Subroutines Called:

OpenWB
MakeInputMatrixSheet
AddCSSheets
MakeOutputMatrixSheet
ClassSubtotals

Called By:

Driver

9. Subroutine: MakeInputMatrixSheet

Purpose:

- Creates the InputMatrix worksheet
- Initializes the InputMatrix worksheet with the contents of CostMatrix

Inputs:

Public Variables:

MaxClass
MaxComps
CostMatrix
MatrixRowOffset
MatrixColOffset

Outputs:

Workbooks:

Creates and initializes worksheet InputMatrix

Subroutines Called:

None

Called By:

MakeWorkingBook

10. Subroutine: FillDKSheet()

Purpose:

- Places component names and numbers on each place-holder (non-Cost Segment) sheet

Inputs:

Public Variables:

CompsMax
ComponentData
iModule
ClassMax
MailClassData

Outputs:

Workbooks:

Component Numbers and Names inserted onto the worksheet

Subroutines Called:

None

Called By:

AddCSSheets

G. Response to User Actions (ModUserInterface)

1. Subroutine: SelectModel()

   Purpose:

   - Take actions when the user selects a new model on the CRA Menu sheet

   Inputs:

   User Selection:

   Model Name

   Outputs:

   Public Variables::

   ModName
   ModPath

   Subroutines Called:

   None

   Called By:

   Event (user selection)

2. Subroutine: ChangeRunType()

   Purpose:

   - Gets the Control file when the user specifies a Rollforward run

   Inputs:

   User Specification:

   Run Type

   Public Variables:

   None

   Outputs:

   Public Variables:

   iRunType

   Subroutines Called:

RFCntlFileBrowse

Called By:

Driver

3. Subroutine: Run()

Purpose:

- Initiates processing when the user clicks the "Run" button from the CRA Menu

Inputs:

Public Variables:

ErrMax

Outputs:

Public Variables:

StartTime
ErrRow
ErrSheet
MenuSheet
ControlSheet
ListSheet
LimitSheet
ErrDat
ynSaveClose

Subroutines Called:

InitializeErrorSheet
CheckPhase1
Driver
RunCleanup
RecordError

Called By:

Event (user click of "Run" button)

4. Subroutine: RunCleanup()

Purpose:

- Finalizes the run.
- Displays a message box giving completion details.

Inputs:

Public Variables:

ErrStop
iRunType

Outputs:

Public Variables:

EndTime

Subroutines Called:

None

Called By:

Run

5. Subroutine: GetRFOptions()

Purpose:

- Gets input information from user when Rollforward option is selected

Inputs:

User specification of file

Public Variables:

SkipClass
MaxClass
MaxComps
MatrixRowOffset
MatrixColOffset

Outputs:

Public Variables:

ModPath
CostMatrix

Subroutines Called:

OpenWB

Called By:

Driver

6. Subroutine: CancelForm()

Purpose:

- Handling action when user clicks "Cancel" on a form

Inputs:

User Input (clicking "Cancel" button)

Outputs:

Public Variables:

FormResult

Subroutines Called:

None

Called By:

Event

7. Subroutine: CancelRFOptions()

Purpose:

- Handles event when user clicks "Cancel" on RF Options form

Inputs:

User Input

Outputs:

Public Variables:

FormResult

Subroutines Called:

None

Called By:

Event

8. Subroutine: EndRFOptions()

Purpose:

- Handles event when user clicks "Finished" on RF Options form

Inputs:

User Input

Form objects

Outputs:

Public Variables:

RateCase
RFYear
Contingency
Rfwd
VerType

Subroutines Called:

None

Called By:

Event

9. Subroutine: RFCntlFileBrowse()

Purpose:

- Gets input file when user specifies Rollforward run type

Inputs:

User Input

Outputs:

Public Variables:

ModPath
CntlBook

Subroutines Called:

None

Called By:

ChangeRunType

H. Utility Functions (ModUtil)

1. Function: IsClass(C)

   Purpose:

   - Determines whether C is a valid mail class
   - Returns True or False

   Inputs:

   Argument, C

   Public Variables:

   MailClassData
   ClassMax

   Outputs:

   Function value (True or False)

   Subroutines Called:

   None

   Called By:

   DataCheck
   CheckRoutine

2. Function: IsComponent(C)

   Purpose:

   - Determines whether C is a valid component
   - Returns True or False

   Inputs:

   Argument, C

   Public Variables:

   ComponentData
   CompsMax

   Outputs:

   Function value (True or False)

   Subroutines Called:

   None

Called By:

DataCheck
CheckRoutine

3. Subroutine: SaveCloseWorkbooks()

Purpose:

- Optionally saves and closes workbooks at the end of a run, as specified by user

Inputs:

Public Variables:

ynSaveClose
SCCount
SaveCloseNames
SaveCloseFlag

Outputs:

Saved / Closed workbooks

Subroutines Called:

None

Called By:

RunCleanup

I. Error Handling Routines (ModErrorHandling)

1. Subroutine: RecordError()

   Purpose:

   - Writes an error on the Errors Sheet in the CRA workbook

   Inputs:

   Public Variables:

   ErrRow
   ErrDat

   Outputs:

   Public Variables:

   ErrMax

   Subroutines Called:

   None

   Called By:

   DataCheck
   NonCompArg
   CheckRoutine
   CheckAVRoutine
   CheckDKA
   CheckDKF
   CheckDKB
   CheckDKRF
   Run
   CheckPhase1
   CheckPhase2
   ReadDistKeys

2. Subroutine: InitializeErrorSheet()

   Purpose:

   - Initializes the Errors Sheet in the CRA workbook

   Inputs:

   Public Variables:

   xlMaxLine

   Outputs:

Public Variables:

ErrRow
ErrMax
ErrStop

Subroutines Called:

None

Called By:

Run

3. Subroutine: DataCheck()

Purpose:

- Main routine for performing data logic / consistency checks

Inputs:

Public Variables:

CRABook
CompsMax
ComponentData
CntlBook
CntlCompSheet
DKMax
DKData
DetailMax
DKDetail

Outputs:

Public Variables:

ErrDat

Subroutines/Functions Called:

IsComponent
IsClass
NonCompArg
CheckRoutine
CheckAVRoutine
CheckDKA
CheckDKF
CheckDKB
CheckDKRF
RecordError

Called By:

Init

4. Subroutine: NonCompArg(iC, Desc)

Purpose:

- Prepares and records the error for an argument that is not a component where iC is the component index and Desc is a text description of the invalid argument

Inputs:

Public Variables:

None

Outputs:

Public Variables:

ErrDat

Subroutines Called:

RecordError

Called By:

DataCheck

5. Subroutine: CheckRoutine(iComp, RType, Arg1, Arg2, Modul)

Purpose:

- Checks that the arguments for the routine are valid where:
  - iComp is the component number
  - RType is the Routine type (e.g., AV, DA, DV, etc)
  - Arg1 and Arg2 are the arguments (usually distribution keys)
  - Modul is the module label (e.g., A, F, B, C)

Inputs:

Public Variables:

ComponentData
CompIndx

Outputs:

Public Variables:

ErrDat

Subroutines/Functions Called:

IsDK
Is983
Is21983
IsComponent
IsClass
RecordError

Called By:

DataCheck

6. Subroutine: CheckAVRoutine(iComp, Arg1, Arg2, Var)

Purpose:

- Checks the validity of input parameters when the AV routine has been specified, where:
  - iComp is the component number
  - Arg1 and Arg2 are the parameters (components) used in the routine
  - Var is the variability

Inputs:

Public Variables:

None

Outputs:

Public Variables:

ErrDat

Subroutines Called:

RecordError

Called By:

DataCheck

7. Function: Is983(C)

Purpose:

- Checks to see whether the argument is a valid component in cost segment 98.3

Inputs:

Public Variables:

CompsMax
ComponentData
CompIndx

Outputs:

Public Variables:

None

Subroutines Called:

None

Called By:

CheckRoutine

8. Function: Is21983(C)

Purpose:

- Checks to see whether the argument is a valid component in either cost segment 21 or cost segment 98.3

Inputs:

Public Variables:

CompsMax
ComponentData
CompIndx

Outputs:

Public Variables:

None

Subroutines Called:

None

Called By:

CheckRoutine

9. Function: IsDK(C)

Purpose:

- Checks to see whether the argument is a valid Distribution Key

Inputs:

Public Variables:

DKMax
DKData

Outputs:

Public Variables:

None

Subroutines Called:

None

Called By:

Phase2
CheckRoutine

10. Subroutine: CheckDKA(iDK)

Purpose:

- Checks to see whether the Distribution Key represented by iDK is referenced by at least one component in Module A

Inputs:

Public Variables:

DKData
CompsMax
ComponentData

Outputs:

Public Variables:

ErrDat

Subroutines Called:

RecordError

Called By:

DataCheck

11. Subroutine: CheckDKF(iDK)

Purpose:

- Checks to see whether the Distribution Key represented by iDK is referenced by at least one component in Module F

Inputs:

Public Variables:

DKData
CompsMax
ComponentData

Outputs:

Public Variables:

ErrDat

Subroutines Called:

RecordError

Called By:

DataCheck

12. Subroutine: CheckDKB(iDK)

Purpose:

- Checks to see whether the Distribution Key represented by iDK is referenced by at least one component in Module B

Inputs:

Public Variables:

DKData
CompsMax
ComponentData

Outputs:

Public Variables:

ErrDat

Subroutines Called:

RecordError

Called By:

DataCheck

VI. VBA PROGRAM CODE

A. Main Program – ModDriver

```
Sub Driver()
'Main control routine for the model

Dim Msg As String
Dim RunStart As Integer

Call Auto_Open
If ErrDat.Level > 0 Then
  ErrStop = True
  Exit Sub
End If

Application.ScreenUpdating = False

Call SelectModel

RunOK = True
Call ChangeRunType

If RunOK = False Then
    Exit Sub
End If

Call Init
If ErrMax > ErrMaxAllow Then
  ErrStop = True
  Exit Sub
End If

If iRunType = 1 Then
  Exit Sub
End If

Windows(CRABook).Activate
```

```
If iRunType = 3 Then
    RunStart = 3
    Call GetRFOptions
    If RunOK = False Then
        Exit Sub
    End If
Else
    RunStart = 1
    Call ReadInputMatrix
End If

Call RptInit
'Check whether error level specification has been exceeded.
If ErrMax > ErrMaxAllow Then
  Msg = "Maximum allowable error level has been exceeded." & vbNewLine & vbNewLine
  Msg = Msg & "You specified the maximum allowable error level at " & ErrMaxAllow & "." & vbNewLine
  Msg = Msg & "The maximum error level encountered during the data checking was " & ErrMax & "." & vbNewLine
  Msg = Msg & vbNewLine & "Do you want to look at the errors page?"
  If MsgBox(Msg, vbYesNo + vbQuestion, "Input error(s) encountered") = vbYes Then
    Windows(CRABook).Activate
    Application.GoTo ("ErrorsSheet")
    ErrStop = True
  End If
  Exit Sub
End If

'iModule is public variable defining the current routine
For iModule = 1 To 5
    Select Case iModule
      Case 1
        'GEDI Module, runs I Report first
        Call Reports
        If iRunType <> 3 Then
            Call MakeWorkingBook
            Call RoutineDriver  'Calls ComponentSubtotals at end
            Call UpdateOutputMatrix
         End If
      Case 2
        'A Module
        If iRunType <> 3 Then
            Call MakeWorkingBook
            Call RoutineDriver  'Calls ComponentSubtotals at end
```

```
            Call UpdateOutputMatrix
            Call Reports
         End If
      Case 3
        'Factor Module
        Call MakeWorkingBook
        Call RoutineDriver  'Calls ComponentSubtotals at end
        Call UpdateOutputMatrix
        Call Reports
      Case 4
        'B Module, runs K Reports also
        'Calculates components needed for D Report
        Call MakeWorkingBook
        Call RoutineDriver  'Calls ComponentSubtotals at end
        Call UpdateOutputMatrix
        Call Reports
        Call KReport
      Case 5
        'C and D Reports only
        Call Reports
        Call DReport
    End Select
Next

Application.Calculation = xlAutomatic
Application.StatusBar = "CRA Model Run Complete"

End Sub
```

B. Initialization Routines – ModInit

```
'*********************************************************************
'This Module contains Initialization routines
'*********************************************************************

Sub Auto_Open()
Dim Msg As String

ErrDat.Level = 0
On Error GoTo AutoOpenErr
Application.GoTo ("VerLabel")
Selection.Value = "Version: 1 (20050314.1645)"
CRABook = ActiveWorkbook.Name
CRAPath = ActiveWorkbook.Path
Application.GoTo ("MenuDisplay")
ActiveWindow.Zoom = True
'Application.Goto ("RunName")
If Right(CRAPath, 1) <> "\" Then
  CRAPath = CRAPath & "\"
End If
Application.GoTo ("MaxError")
On Error GoTo 0
If ErrDat.Level > 0 Then
  Exit Sub
End If

'The following sub consists only of a series of assignment statements.
'No error can occur in it.
Call SetRowColVariables
LoaderPath = CRAPath

ActiveWindow.DisplayHeadings = False
'Turn off the formula bar
'Application.DisplayFormulaBar = False
ActiveWindow.WindowState = xlMaximized
Application.CommandBars("Reviewing").Visible = False
Application.CommandBars("Drawing").Visible = False

Exit Sub
```

```
AutoOpenErr:
Msg = "Basic initialization failed." & vbCrLf & vbCrLf _
    & "One or more named ranges on the Menu page are missing."
MsgBox Msg, vbOKOnly + vbCritical, "Initialization failure"
ErrDat.Level = 3


End Sub
```

```
Sub ReadLoaderModelData()
'Reads the name and path of the currently-selected model from the Data Entry Module
'Copies the entire list of available model names and paths from the Data Entry Module to the CRA workbook

'MRow is the next available row when copying data into this book
'LRow is the next row in Loader to be copied
Dim MRow, LRow, R As Long
Dim MRFirst As Long
Dim MC, LC As Integer
Dim CRAModels, LModels As String
Dim Rng As String
Dim Result As Integer

'Reads data in this book that describes how Loader is set up
Call LoaderLayout

'First clear out anything that's already in this book
Windows(CRABook).Activate
Application.GoTo ("ModelsSheet")
CRAModels = ActiveSheet.Name
Cells(1, 1).Select
Selection.End(xlDown).Select
Rng = "A2:B" & Trim(Str(Selection.Row))
Range(Rng).Select
Selection.ClearContents
MRow = 2
MC = 1
MRFirst = 2

'Copy data from Loader workbook
SCFlag = False
Call OpenWB(LoaderName, LoaderPath, Result)
Application.GoTo (LoaderModelsRange)
LModels = ActiveSheet.Name
LRow = Selection.Row + 1
LC = Selection.Column
Cells(LRow, LC).Select

Do Until Trim(Selection.Value) = ""
  ModName = Selection.Value
  Cells(LRow, LC + 1).Select
  ModPath = Selection.Value
```

```
  If Right(ModPath, 1) <> "\" Then ModPath = ModPath & "\"
  Windows(CRABook).Activate
  Cells(MRow, MC).Select
  Selection.Value = ModName
  Cells(MRow, MC + 1).Select
  Selection.Value = "'" & ModPath
  MRow = MRow + 1
  Windows(LoaderName).Activate
  LRow = LRow + 1
  Cells(LRow, LC).Select
Loop
Windows(CRABook).Activate
Rng = ActiveSheet.Name & "!$A$" & Trim(Str(MRFirst)) & ":$B$" & Trim(Str(MRow - MRFirst + 1))
Application.GoTo ("ModelName")
ActiveSheet.Shapes("Drop Down 8").Select
Selection.ListFillRange = Rng

End Sub
```

```
Sub Init()
Dim R1 As Long
Dim I As Integer
Dim Msg As String

Application.StatusBar = "Initializing internal indexes"

If iRunType <> 3 Then
    Call ReadRunData
End If

'Clear out everything related to files that have to be saved and closed
ReDim SaveCloseNames(0)
ReDim SaveCloseFlag(0)
SCCount = 0

'Read maximum allowable error level
Application.GoTo ("MaxError")
ErrMaxAllow = Selection.Value

'Read variables that are used to dimension arrays
Application.GoTo reference:="LimitsSheet"
R1 = Selection.Row
Cells(R1 + 1, 1).Select
MaxClass = Selection.Value
Cells(R1 + 2, 1).Select
MaxCS = Selection.Value
Cells(R1 + 3, 1).Select
MaxCompon = Selection.Value
Cells(R1 + 4, 1).Select
MaxDKComp = Selection.Value
Cells(R1 + 5, 1).Select
MaxComps = Selection.Value
Cells(R1 + 6, 1).Select
MaxDefs = Selection.Value
Cells(R1 + 7, 1).Select
TotVVClass = Selection.Value
Cells(R1 + 8, 1).Select
OtherClass = Selection.Value
Cells(R1 + 9, 1).Select
TotClass = Selection.Value
Cells(R1 + 10, 1).Select
```

```
SkipClass = Selection.Value
Cells(R1 + 11, 1).Select
LastCS = Selection.Value
'ReDim the arrays whose limits have just been read in
ReDim ComponentData(MaxCompon)
ReDim CostMatrix(MaxClass, MaxComps)
Call InitMtxChange
ReDim CompColumn(MaxComps)
ReDim CompSheet(MaxComps)
ReDim CompIndx(MaxComps)
ReDim CSIndex(MaxCS)
CSCount = 0

'Read info on how the Loader is set up
Call LoaderLayout

Application.StatusBar = "Reading Input Table Columns"
Call ReadInputTableColumns

Call CheckPhase2
If ErrMax > 1 Then
  Msg = "Initialization error(s) encountered." & vbCrLf & vbCrLf _
      & "One or more required workbooks/sheets not found.  See Errors sheet." & vbCrLf & vbCrLf _
      & "Cannot continue."
  MsgBox Msg, vbOKOnly + vbCritical, "Initialization error"
  Exit Sub
End If

Application.StatusBar = "Reading Control Table"
Call ReadControlTable
Call InitExtraSegments

Application.StatusBar = "Reading Mail Classes from Data Entry Module"
'Mail Class Data
Call ClassArrayInit
Call RevArrayInit
Call ReadDRptData

Call DataCheck

End Sub
```

```
Sub CheckPhase1()
'This is the first level of error checking.
'It looks to see that all the named ranges used in this workbook are present.
'Any error encountered here is a fatal (level 3) error.

Dim NRange As String

On Error GoTo Phase1Err
NRange = "ModelName"
Application.GoTo (NRange)
NRange = "SaveClose"
Application.GoTo (NRange)
NRange = "RunType"
Application.GoTo (NRange)
NRange = "MaxError"
Application.GoTo (NRange)
NRange = "USPSorPRC"
Application.GoTo (NRange)
NRange = "MenuDisplay"
Application.GoTo (NRange)
NRange = "VerLabel"
Application.GoTo (NRange)
NRange = "ErrorsSheet"
Application.GoTo (NRange)
NRange = "InputTableSheet"
Application.GoTo (NRange)
NRange = "InputTableLayout"
Application.GoTo (NRange)
NRange = "CntlBookArea"
Application.GoTo (NRange)
NRange = "LoaderBook"
Application.GoTo (NRange)
NRange = "DKMasterSheet"
Application.GoTo (NRange)
NRange = "DKAddendsSheet"
Application.GoTo (NRange)
NRange = "ListsSheet"
Application.GoTo (NRange)
NRange = "USPSOptions"
Application.GoTo (NRange)
NRange = "ModelsSheet"
Application.GoTo (NRange)
```

```
NRange = "LimitsSheet"
Application.GoTo (NRange)
NRange = "ReportSheet"
Application.GoTo (NRange)

Exit Sub

Phase1Err:
'For error encountered while looking for named ranges
ErrDat.Book = ActiveWorkbook.Name
ErrDat.Desc = "Named range cannot be found: " & NRange
ErrDat.Level = 3
If ErrDat.Sheet = "" Then
  ErrDat.Sheet = "(Unknown Worksheet)"
End If
Call RecordError
ErrDat.Sheet = ""
Resume Next

End Sub
```

```
Sub CheckPhase2()
'Second level of error checking.
'This is run after data from the CRA Workbook has already been read.
'(Called near the end of Sub Init.)
'It looks to see that all the workbooks needed are present.
'It looks to see that all the worksheets in the workbooks are present.

Dim WBook As String
Dim WSheet As String
Dim NRange As String
Dim EFlag As Boolean
Dim Result As Integer
Dim FullName As String

'See whether each required workbook is available

'Loader model
On Error GoTo Phase2Err1
WBook = LoaderName
Call OpenWB(WBook, LoaderPath, Result)
If Result > 1 Then
  ActiveWorkbook.Close
  Workbooks(CRABook).Activate
  Error (9999)
  FullName = LoaderPath & WBook
  On Error GoTo 0
'Deletes an accidentally-created workbook
  Kill FullName
Else
'Look for models sheet
  On Error GoTo Phase2Err2
  WSheet = LoaderModelsSheet
  Sheets(WSheet).Activate
'Look for each named range in the Loader model
  On Error GoTo Phase2Err5
  Workbooks(LoaderName).Activate
  NRange = LoaderClassRange
  Application.GoTo (NRange)
  NRange = LoaderCurrentModel
  Application.GoTo (NRange)
  NRange = LoaderCurrentDir
  Application.GoTo (NRange)
```

```
  NRange = LoaderModelsRange
  Application.GoTo (NRange)
  NRange = LoaderRevRange
  Application.GoTo (NRange)
End If

'Cntl book
On Error GoTo Phase2Err1
WBook = CntlBook
Call OpenWB(WBook, ModPath, Result)
If Result > 1 Then
  ActiveWorkbook.Close
  Workbooks(CRABook).Activate
  Error (9999)
  FullName = ModPath & WBook
  On Error GoTo 0
'Deletes an accidentally-created workbook
  Kill FullName
Else
  On Error GoTo Phase2Err2
  Workbooks(CntlBook).Activate
  WSheet = CntlCompSheet
  Sheets(WSheet).Activate
  WSheet = CntlDKSheet
  Sheets(WSheet).Activate
  WSheet = CntlDKDetailSheet
  Sheets(WSheet).Activate
End If
On Error GoTo 0

'I file - only if not a Roll Forward run
If iRunType <> 3 Then
  On Error GoTo Phase2Err1
  WBook = ModName & ".I.xls"
  Call OpenWB(WBook, ModPath, Result)
  If Result > 1 Then
    ActiveWorkbook.Close
    Workbooks(CRABook).Activate
    Error (9999)
    FullName = ModPath & WBook
    On Error GoTo 0
'Deletes an accidentally-created workbook
```

```
    Kill FullName
  Else
'Look for InputMatrix sheet
    EFlag = False
    Workbooks(WBook).Activate
    WSheet = IFileSheetName
    On Error GoTo Phase2Err3
    Sheets(WSheet).Activate
    If EFlag Then
'If there is only one sheet in this book, we can assume it is the one we want - level 1 error.
'If more than one sheet, then this is a level 3 error.
      If Sheets.Count = 1 Then
        IFileSheetName = ActiveSheet.Name
        On Error GoTo Phase2Err4
        Error (9999)
      Else
        On Error GoTo Phase2Err2
        Error (9999)
      End If
    End If
  End If
End If
On Error GoTo 0

Workbooks(CRABook).Activate

Exit Sub

'******************************
'Error-handling routines below
'******************************
Phase2Err1:
'For error encountered while trying to open a required workbook
ErrDat.Book = WBook
ErrDat.Desc = "Specified workbook does not exist"
ErrDat.Level = 3
ErrDat.Sheet = "(All sheets)"
Call RecordError
ErrDat.Sheet = ""
Resume Next

Phase2Err2:
```

```
'For error encountered looking for worksheet in a workbook
ErrDat.Book = WBook
ErrDat.Desc = "Worksheet cannot be found."
ErrDat.Level = 3
ErrDat.Sheet = WSheet
Workbooks(CRABook).Activate
Call RecordError
ErrDat.Sheet = ""
Workbooks(WBook).Activate
Resume Next

Phase2Err3:
'For error encountered looking for InputMatrix sheet - potentially recoverable
EFlag = True
Resume Next

Phase2Err4:
'For error encountered looking for InputMatrix sheet, assume only sheet in book is the one
ErrDat.Book = WBook
ErrDat.Desc = "Sheet '" & WSheet & "' not found.  Assuming correct sheet is '" & IFileSheetName & "'"
ErrDat.Level = 1
ErrDat.Sheet = IFileSheetName
Workbooks(CRABook).Activate
Call RecordError
Workbooks(WBook).Activate
ErrDat.Sheet = ""
Resume Next

Phase2Err5:
'For error encountered looking for a named range in the Loader workbook
ErrDat.Book = LoaderName
ErrDat.Sheet = "Named range"
ErrDat.Level = 3
ErrDat.Row = 0
ErrDat.Desc = "Loader module named range (" & NRange & ") not found"
Call RecordError

End Sub
```

```
Sub ReadControlTable()
Call ReadDistribution
Call ReadDistKeys
Call ReadDKDocumentation
End Sub
Sub ReadDistribution()
'Assumes the Control workbook is already open
'Determines where the last line of data is, then goes back to the top and
'keeps reading until it gets to the last line.

Dim LastRow As Long
Dim R As Long
Dim CID, C As Long
Dim Result As Integer
Dim Pt As Integer
Dim I, J As Integer
Dim PFlag As Boolean

SCFlag = False
Call OpenWB(CntlBook, ModPath, Result)
Workbooks(CntlBook).Activate
Sheets(CntlCompSheet).Activate

Cells(xlMaxLine, NumberCol).Select
Selection.End(xlUp).Select
LastRow = Selection.Row
C = 0
For R = CompMasterFirstDataRow To LastRow
  C = C + 1
  Cells(R, NumberCol).Select
  If Len(Trim(Selection.Value)) > 0 And IsNumeric(Selection.Value) Then
    CID = Selection.Value
    If CID > 0 And CID <= MaxComps Then
      CompIndx(CID) = C
      ComponentData(C).Label = Right("0000" & Trim(Str(CID)), 4)
      Cells(R, TitleCol).Select
      ComponentData(C).Title = Trim(Selection.Value)
      Cells(R, SubSegmentCol).Select
      ComponentData(C).SubSegment = Trim(Selection.Value)
'Derive the Cost Segment
'CS is the portion to the left of the first decimal point if CS <= LastCS
'CS is the whole SubSegment if CS > LastCS
```

```
ComponentData(C).CS = Val(ComponentData(C).SubSegment)
If ComponentData(C).CS < (LastCS + 1) Then
  If Len(ComponentData(C).SubSegment) > 0 Then
    Pt = InStr(ComponentData(C).SubSegment, ".")
    If Pt > 0 Then
      ComponentData(C).CS = Val(Mid(ComponentData(C).SubSegment, 1, Pt - 1))
    Else
      ComponentData(C).CS = Val(ComponentData(C).SubSegment)
    End If
  ElseIf UCase(Left(ComponentData(C).Title, 3)) = "C/S" Then
    ComponentData(C).CS = Val(Trim(Mid(ComponentData(C).Title, 5, 2)))
  End If
End If
Cells(R, SourceCol).Select
ComponentData(C).Source = Selection.Value
Cells(R, LibRefCol).Select
ComponentData(C).LibRef = Selection.Value
Cells(R, InputTypeCol).Select
ComponentData(C).InputType = Selection.Value
Cells(R, ManualVariabilityCol).Select
ComponentData(C).ManualVariability = Val(Trim(Selection.Value))
Cells(R, PrintICol).Select
ComponentData(C).PrintI = Trim(Selection.Value)
Cells(R, Module1Col).Select
ComponentData(C).Module1 = Trim(Selection.Value)
Cells(R, Distrib1Col).Select
ComponentData(C).Distrib1 = Val(Trim(Selection.Value))
Cells(R, Misc1Col).Select
ComponentData(C).Misc1 = Val(Trim(Selection.Value))
Cells(R, ModuleACol).Select
ComponentData(C).ModuleA = Trim(Selection.Value)
Cells(R, DistribACol).Select
ComponentData(C).DistribA = Val(Trim(Selection.Value))
Cells(R, MiscACol).Select
ComponentData(C).MiscA = Val(Trim(Selection.Value))
Cells(R, PrintACol).Select
ComponentData(C).PrintA = Trim(Selection.Value)
Cells(R, ParentACol).Select
ComponentData(C).ParentA = Val(Trim(Selection.Value))
Cells(R, ModuleFCol).Select
ComponentData(C).ModuleF = Trim(Selection.Value)
Cells(R, DistribFCol).Select
```

```
      ComponentData(C).DistribF = Val(Trim(Selection.Value))
      Cells(R, MiscFCol).Select
      ComponentData(C).MiscF = Val(Trim(Selection.Value))
      Cells(R, PrintFCol).Select
      ComponentData(C).PrintF = Trim(Selection.Value)
      Cells(R, ParentFCol).Select
      ComponentData(C).ParentF = Val(Trim(Selection.Value))
      Cells(R, ModuleBCol).Select
      ComponentData(C).ModuleB = Trim(Selection.Value)
      Cells(R, DistribBCol).Select
      ComponentData(C).DistribB = Val(Trim(Selection.Value))
      Cells(R, MiscBCol).Select
      ComponentData(C).MiscB = Val(Trim(Selection.Value))
      Cells(R, PrintBCol).Select
      ComponentData(C).PrintB = Trim(Selection.Value)
      Cells(R, ParentBCol).Select
      ComponentData(C).ParentB = Val(Trim(Selection.Value))
      Cells(R, PrintCCol).Select
      ComponentData(C).PrintC = Trim(Selection.Value)
      Cells(R, ParentCCol).Select
      ComponentData(C).ParentC = Val(Trim(Selection.Value))
      ComponentData(C).Row = R
    Else
'This is an error condition of some kind, numeric component outside of range
    End If
  Else
'Check to see if we're dealing with a Cost Segment
    Cells(R, TitleCol).Select
    ComponentData(C).Title = Trim(Selection.Value)
    If UCase(Left(Trim(Selection.Value), 3)) = "C/S" Then
      CSCount = CSCount + 1
      ReDim Preserve CostSegment(CSCount)
      CostSegment(CSCount).CSNumber = Val(Trim(Mid(Selection.Value, 5, 2)))
      CostSegment(CSCount).CSName = Trim(Mid(Selection.Value, 7))
      CSIndex(CostSegment(CSCount).CSNumber) = CSCount
    End If
  End If
Next

CSMax = CSCount
CompsMax = C
```

```
'Calculate which cost segments are subtotals (i.e., they have children)
For I = 1 To CompsMax
  PFlag = False
  If Val(ComponentData(I).Label) > 0 Then
    For J = 1 To CompsMax
      If Val(ComponentData(J).ParentA) = Val(ComponentData(I).Label) Then
        ComponentData(I).SubTotalA = True
      End If
      If Val(ComponentData(J).ParentF) = Val(ComponentData(I).Label) Then
        ComponentData(I).SubTotalF = True
      End If
      If Val(ComponentData(J).ParentB) = Val(ComponentData(I).Label) Then
        ComponentData(I).SubTotalB = True
      End If
      If Val(ComponentData(J).ParentC) = Val(ComponentData(I).Label) Then
        ComponentData(I).SubTotalC = True
      End If
    Next
  End If
Next

End Sub
```

```
Sub ReadDistKeys()
'Assumes the Control workbook is already open
'Determines where the last line of data is, then goes back to the top and
'keeps reading until it gets to the last line.

Dim LastRow As Long
Dim R As Long
Dim CID, C As Long
ReDim DKData(MaxDefs)
ReDim DKIndx(MaxComps)

Workbooks(CntlBook).Activate
Sheets(CntlDKSheet).Activate

Cells(xlMaxLine, DKCompCol).Select
Selection.End(xlUp).Select
LastRow = Selection.Row
C = 0
For R = 2 To LastRow
  Cells(R, DKCompCol).Select
'This does not use the IsComponent function because some DKs might not be Components
  If IsNumeric(Selection.Value) Then
    CID = Selection.Value
    If CID > 0 And CID <= MaxComps Then
      C = C + 1
      DKMax = C
      DKIndx(CID) = C
      DKData(C).DKComp = CID
      Cells(R, DKTitleCol).Select
      DKData(C).DKTitle = Selection.Value
      Cells(R, DKContentsCol).Select
      DKData(C).DKContents = Selection.Value
      Cells(R, DKAffectedCol).Select
      DKData(C).DKAffected = Selection.Value
      Cells(R, DKAddendsCol).Select
      DKData(C).DKAddends = False
      If UCase(Trim(Selection.Value)) = "X" Then
        DKData(C).DKAddends = True
      End If
      Cells(R, DKACol).Select
      DKData(C).DKA = False
      If UCase(Trim(Selection.Value)) = "X" Then
```

```
        DKData(C).DKA = True
      End If
      Cells(R, DKFCol).Select
      DKData(C).DKF = False
      If UCase(Trim(Selection.Value)) = "X" Then
        DKData(C).DKF = True
      End If
      Cells(R, DKBCol).Select
      DKData(C).DKB = False
      If UCase(Trim(Selection.Value)) = "X" Then
        DKData(C).DKB = True
      End If
      Cells(R, DKRFCol).Select
      DKData(C).DKRF = False
      If UCase(Trim(Selection.Value)) = "X" Then
        DKData(C).DKRF = True
      End If
    Else
      ErrDat.Book = CntlBook
      ErrDat.Sheet = CntlDKSheet
      ErrDat.Row = R
      ErrDat.Level = 2
      ErrDat.Desc = "DK/Component number out of allowable range"
      Call RecordError
    End If
  Else
    ErrDat.Book = CntlBook
    ErrDat.Sheet = CntlDKSheet
    ErrDat.Row = R
    ErrDat.Level = 2
    ErrDat.Desc = "Non-numeric entry where DK/Component number was expected"
    Call RecordError
  End If
Next

End Sub
```

```
Sub ReadDKDocumentation()
'Assumes the Control workbook is already open
'Determines where the last line of data is, then goes back to the top and
'keeps reading until it gets to the last line.

Dim LastRow As Long
Dim R As Long
Dim iCol As Integer
Dim ACnt, KCnt As Integer

Workbooks(CntlBook).Activate
Sheets(CntlDKDetailSheet).Activate

Cells(xlMaxLine, DKACompCol).Select
Selection.End(xlUp).Select
LastRow = Selection.Row
iCol = DKADataCol

Cells(1, iCol).Select
Do Until Len(Trim(Selection.Value)) = 0
'Read header info
  KCnt = KCnt + 1
  ReDim Preserve DKDetail(KCnt)
  DKDetail(KCnt).DKATitle = Selection.Value
  Cells(DKAKeyRow, iCol).Select
'Add an error check here
  DKDetail(KCnt).DKNumber = Selection.Value
  Cells(DKAAddendCountRow, iCol).Select
  DKDetail(KCnt).AddendCount = Selection.Value
'This will count the actual number of addends to compare to the header
  ACnt = 0
  For R = DKAFirstDataRow To LastRow
    Cells(R, iCol).Select
    If UCase(Trim(Selection.Value)) = "X" Then
      Cells(R, 2).Select
      If IsNumeric(Selection.Value) Then
        ACnt = ACnt + 1
'In the future we want to see whether we can dynamically resize Addends(ACnt) here
        DKDetail(KCnt).Addends(ACnt) = Selection.Value
      End If
    End If
  Next
```

```
  If ACnt <> DKDetail(KCnt).AddendCount Then
    ErrDat.Book = CntlBook
    ErrDat.Sheet = CntlDKDetailSheet
    ErrDat.Row = DKAAddendCountRow
    ErrDat.Level = 2
    ErrDat.Desc = "Actual addend count for DK " & DKDetail(KCnt).DKNumber & " (" & ACnt & ") does not match number in
workbook (" & DKDetail(KCnt).AddendCount & ")"
  End If
  iCol = iCol + 1
  Cells(1, iCol).Select
Loop
DetailMax = KCnt

End Sub
```

```
Sub InitExtraSegments()
'Looks through the SubSegment data to determine which Subsegments are to be
'treated like Cost Segments with respect to setting up worksheets, etc.

Dim I, xI As Integer
Dim Lx As Single
Dim FirstKey As String
Dim LastLx As Single

xI = 0
Lx = 0
FirstKey = "Calculated "
LastLx = 0

For I = 1 To CompsMax
  If ComponentData(I).CS >= (LastCS + 1) Then
    If ComponentData(I).SubSegment <> Lx And ComponentData(I).SubSegment <> LastLx Then
      xI = xI + 1
      Lx = ComponentData(I).SubSegment
      LastLx = Lx
      ReDim Preserve PlaceHolderSheets(xI)
      ReDim Preserve PlaceHolderTypes(xI)
      PlaceHolderSheets(xI) = "CS" & Trim(Str(Lx))
      PlaceHolderTypes(xI) = "Cost Segment " & Trim(Str(Lx)) & ": " & ComponentData(I - 1).Title
      FirstKey = ""
    End If
  End If
Next

xI = xI + 1
ReDim Preserve PlaceHolderSheets(xI)
ReDim Preserve PlaceHolderTypes(xI)
PlaceHolderSheets(xI) = "WorkArea"
PlaceHolderTypes(xI) = "Work Area"
PlaceHolderCount = xI

End Sub
```

```
Sub ReadRunData()
'This sub will read data from the Models sheet
Dim RunID As Integer

'First get the correct run number from the Menu sheet
Application.GoTo ("ModelName")
RunID = Selection.Value
'Go to the Models sheet and find the path for this run
Application.GoTo ("ModelsSheet")
Cells(RunID + 1, 2).Select
ModPath = Selection.Value
If Right(ModPath, 1) <> "\" Then ModPath = ModPath & "\"
Cells(RunID + 1, 1).Select
ModName = Selection.Value

End Sub
```

```
Sub ReadInputTableColumns()
'Reads the columns to expect each data element in the input table
Dim R As Long
Dim C As Integer

'This statement is just to get us to the right worksheet
Application.GoTo ("InputTableLayout")
Cells(2, 2).Select
NumberCol = Selection.Value
Cells(3, 2).Select
TitleCol = Selection.Value
Cells(4, 2).Select
SubSegmentCol = Selection.Value
Cells(5, 2).Select
SourceCol = Selection.Value
Cells(6, 2).Select
LibRefCol = Selection.Value
Cells(7, 2).Select
InputTypeCol = Selection.Value
Cells(8, 2).Select
ManualVariabilityCol = Selection.Value
Cells(9, 2).Select
PrintICol = Selection.Value
Cells(10, 2).Select
Module1Col = Selection.Value
Cells(11, 2).Select
Distrib1Col = Selection.Value
Cells(12, 2).Select
Misc1Col = Selection.Value
Cells(13, 2).Select
ModuleACol = Selection.Value
Cells(14, 2).Select
DistribACol = Selection.Value
Cells(15, 2).Select
MiscACol = Selection.Value
Cells(16, 2).Select
PrintACol = Selection.Value
Cells(17, 2).Select
ParentACol = Selection.Value
Cells(18, 2).Select
ModuleFCol = Selection.Value
Cells(19, 2).Select
```

```
DistribFCol = Selection.Value
Cells(20, 2).Select
MiscFCol = Selection.Value
Cells(21, 2).Select
PrintFCol = Selection.Value
Cells(22, 2).Select
ParentFCol = Selection.Value
Cells(23, 2).Select
ModuleBCol = Selection.Value
Cells(24, 2).Select
DistribBCol = Selection.Value
Cells(25, 2).Select
MiscBCol = Selection.Value
Cells(26, 2).Select
PrintBCol = Selection.Value
Cells(27, 2).Select
ParentBCol = Selection.Value
Cells(28, 2).Select
PrintCCol = Selection.Value
Cells(29, 2).Select
ParentCCol = Selection.Value
Cells(30, 2).Select
CompMasterFirstDataRow = Selection.Value

Application.GoTo ("CntlBookArea")
R = Selection.Row
C = Selection.Column
If iRunType <> 3 Then
  CntlBook = ModName & ".CNTL.xls"
End If

R = R + 1
Cells(R, C).Select
CntlCompSheet = Trim(Selection.Value)
R = R + 1
Cells(R, C).Select
CntlDKSheet = Trim(Selection.Value)
R = R + 1
Cells(R, C).Select
CntlDKDetailSheet = Trim(Selection.Value)

'D REPORT
```

```
R = R + 1
Cells(R, C).Select
CntlDReportSheet = Trim(Selection.Value)

Application.GoTo ("DKMasterSheet")
R = Selection.Row
C = Selection.Column + 1
Cells(R, C).Select
DKCompCol = Selection.Value
Cells(R + 1, C).Select
DKTitleCol = Selection.Value
Cells(R + 2, C).Select
DKContentsCol = Selection.Value
Cells(R + 3, C).Select
DKAffectedCol = Selection.Value
Cells(R + 4, C).Select
DKAddendsCol = Selection.Value
Cells(R + 5, C).Select
DKACol = Selection.Value
Cells(R + 6, C).Select
DKFCol = Selection.Value
Cells(R + 7, C).Select
DKBCol = Selection.Value
Cells(R + 8, C).Select
DKRFCol = Selection.Value

Application.GoTo ("DKAddendsSheet")
R = Selection.Row
C = Selection.Column + 1
Cells(R, C).Select
DKACompCol = Selection.Value
Cells(R + 1, C).Select
DKATitleCol = Selection.Value
Cells(R + 2, C).Select
DKADataCol = Selection.Value
Cells(R + 3, C).Select
DKAFirstDataRow = Selection.Value
Cells(R + 4, C).Select
DKAKeyRow = Selection.Value
Cells(R + 5, C).Select
DKAAddendCountRow = Selection.Value
```

```
Application.GoTo ("IFileSheet")
R = Selection.Row
C = Selection.Column
Cells(R, C).Select
IFileSheetName = Trim(Selection.Value)

'D REPORT
Application.GoTo ("DReportSheet")
R = Selection.Row
C = Selection.Column + 1
Cells(R, C).Select
DCompCol = Selection.Value
Cells(R + 1, C).Select
DRoutineCol = Selection.Value
Cells(R + 2, C).Select
DParm1Col = Selection.Value
Cells(R + 3, C).Select
DParm2Col = Selection.Value
Cells(R + 4, C).Select
DParm3Col = Selection.Value
Cells(R + 5, C).Select
DPrintCol = Selection.Value
Cells(R + 6, C).Select
DRFOnlyCol = Selection.Value

End Sub
```

```
Sub LoaderLayout()
'Reads data about how the Data Entry Module is set up
Dim R As Long
Dim C As Integer

Windows(CRABook).Activate
'Note: LoaderBook is a named range in this workbook that describes the layout of the Loader workbook
Application.GoTo ("LoaderBook")
R = Selection.Row
LoaderName = Trim(Selection.Value)
R = R + 1
Cells(R, 1).Select
LoaderModelsSheet = Trim(Selection.Value)
'Note: Next four variables are named ranges in the Loader workbook
R = R + 1
Cells(R, 1).Select
LoaderClassRange = Trim(Selection.Value)
R = R + 1
Cells(R, 1).Select
LoaderCurrentModel = Trim(Selection.Value)
R = R + 1
Cells(R, 1).Select
LoaderCurrentDir = Trim(Selection.Value)
R = R + 1
Cells(R, 1).Select
LoaderModelsRange = Trim(Selection.Value)
R = R + 1
Cells(R, 1).Select
LoaderRevRange = Trim(Selection.Value)

End Sub
```

```
Sub ClassArrayInit()

Dim I, J, R As Long
Dim Result As Integer
ReDim ClassRow(MaxClass)

'Initializes the Mail CLasses array

Application.StatusBar = "Reading Structure of Mail Class data"

'Go to the Loader workbook and read in the Mail Class data
Call OpenWB(LoaderName, CRAPath, Result)
Application.GoTo (LoaderClassRange)

'This reads the entire range into the MailClass data structure
R = Selection.Row
I = 1
Do Until Trim(Selection.Value) = ""
  ReDim Preserve MailClassData(I)
  ReDim Preserve ClassIndx(I)
  For J = 1 To 8
    Cells(R, J).Select
    Select Case J
      Case 1
        MailClassData(I).Name = Selection.Value
      Case 2
        MailClassData(I).Number = Selection.Value
        ClassIndx(I) = MailClassData(I).Number
      Case 3
        MailClassData(I).Level = Selection.Value
      Case 4
        MailClassData(I).Parent = Selection.Value
      Case 5
        MailClassData(I).SubTotal = Selection.Value
      Case 6
        MailClassData(I).Display1 = Selection.Value
      Case 7
        MailClassData(I).Display2 = Selection.Value
      Case 8
        MailClassData(I).Border = Selection.Value
    End Select
  Next
```

```
  ClassRow(MailClassData(I).Number) = FirstRow + I - 1
  R = R + 1
  I = I + 1
  Cells(R, 1).Select
Loop
ClassMax = I - 1
Exit Sub

'Note that error checking is not yet programmed
CAInitErr:
ErrFlag = 1
Resume Next

End Sub
```

```
Public Sub ReadInputMatrix()
'Reads in the data from the IMatrix to start the whole process

Dim IBookName As String
Dim Result As Integer
Dim C As Integer
Dim R As Long

IBookName = ModName & ".I.xls"
SCFlag = False
Call OpenWB(IBookName, ModPath, Result)
Sheets("InputMatrix").Activate
For C = SkipClass To MaxClass
  Application.StatusBar = "Reading Input Matrix, class " & C & " of " & MaxClass
  For R = 1 To MaxComps
    Cells(R + MatrixRowOffset, C + MatrixColOffset).Select
    CostMatrix(C, R) = Selection.Value
  Next
Next
End Sub
```

```
Public Sub SetRowColVariables()
'This sub sets the values of the Row and Column header variables and other "constants"
'All the variables are declared as Public in ModPublicVariables
HeadRow = 1
CompNameRow = 2
CompNoRow = 3
DocRow = 4
VariabilityRow = 5
ClassLabelRow = 6
FirstRow = ClassLabelRow + 1
SeqCol = 1
ClassNameCol = 2
ClassNoCol = 3
ColHeadCol = 3
FirstColumn = 4
MatrixRowOffset = 3
MatrixColOffset = 3
ModuleLabel(1) = "Mod1"
ModuleLabel(2) = "A"
ModuleLabel(3) = "F"
ModuleLabel(4) = "B"
ModuleLabel(5) = "C"

End Sub
```

```
Public Sub InitMtxChange()
'Initializes the array MtxChange to all false
Dim J As Integer

ReDim MtxChange(MaxComps)

For J = 1 To MaxComps
  MtxChange(J) = False
Next

End Sub
```

```
Sub ReadDRptData()
'Initializes the DData array

Workbooks(CntlBook).Activate
Sheets(CntlDReportSheet).Activate

'This reads the D Report data into the array
Cells(HeadRow + 1, DCompCol).Select
R = Selection.Row
I = 1
Do Until Trim(Selection.Value) = ""
    ReDim Preserve DData(I)
    DData(I).DComp = Selection.Value
    Cells(R, DRoutineCol).Select
    DData(I).DRoutine = Selection.Value
    Cells(R, DParm1Col).Select
    DData(I).DParm1 = Selection.Value
    Cells(R, DParm2Col).Select
    DData(I).DParm2 = Selection.Value
    Cells(R, DParm3Col).Select
    DData(I).DParm3 = Selection.Value
    Cells(R, DPrintCol).Select
    DData(I).DPrint = Trim(Selection.Value)
    Cells(R, DRFOnlyCol).Select
    DData(I).DRFOnly = Trim(Selection.Value)
    R = R + 1
    I = I + 1
    Cells(R, DCompCol).Select
Loop
DMax = I - 1
Exit Sub

'Note that error checking is not yet programmed
DInitErr:
ErrFlag = 1
Resume Next

End Sub
```

```
Sub RevArrayInit()

Dim I, J, R As Long
Dim Result As Integer

'Initializes the Revenue Classes array

Application.StatusBar = "Reading Structure of Revenue Class data"

'Go to the Loader workbook and read in the Mail Class data
Call OpenWB(LoaderName, CRAPath, Result)
Application.GoTo (LoaderRevRange)

'This reads the entire range into the RevenueClass data structure
R = Selection.Row
I = 1
Do Until Trim(Selection.Value) = ""
  ReDim Preserve RevClassData(I)
  For J = 1 To 8
    Cells(R, J).Select
    Select Case J
      Case 1
        RevClassData(I).Name = Selection.Value
      Case 2
        RevClassData(I).Number = Selection.Value
      Case 3
        RevClassData(I).Level = Selection.Value
      Case 4
        RevClassData(I).Parent = Selection.Value
      Case 5
        RevClassData(I).SubTotal = Selection.Value
      Case 6
        RevClassData(I).Display1 = Selection.Value
      Case 7
        RevClassData(I).Display2 = Selection.Value
      Case 8
        RevClassData(I).Border = Selection.Value
    End Select
  Next
  R = R + 1
  I = I + 1
  Cells(R, 1).Select
```

```
Loop
RevMax = I - 1
Exit Sub

'Note that error checking is not yet programmed
CAInitErr:
ErrFlag = 1
Resume Next

End Sub
```

C. Calculations – ModRoutines

```
Sub RoutineDriver()
'Main driving subroutine for writing formulas to do CRA routines

'Public variable iModule is the index of the module currently being run
'Assumes current module workbook is already open and active
Application.StatusBar = ModuleLabel(iModule) & " Workbook: Writing Routine Formulas"
WorkAreaCol = 0

Call Phase1
    'AV routine

Call Phase2
    'DA, DV, DX, DZ, CV routines

Call Phase3
    'CP routine

Call Phase4
    'ST, SA, AJ routines

Call ComponentSubtotals

If iModule = 4 Then
    Call RunDRoutines
End If

End Sub

'Calls the individual routines
```

```
Public Sub Phase1()
'Calls all the Phase 1 routines (AV)

'Go through the ComponentData array for the current module, and execute any AV routines
Dim iComp As Integer
Dim ThisComp As Integer
Dim ThisRoutine As String

For iComp = 1 To CompsMax
    If iModule = 1 Then
        ThisRoutine = ComponentData(iComp).Module1
    ElseIf iModule = 2 Then
        ThisRoutine = ComponentData(iComp).ModuleA
    ElseIf iModule = 3 Then
        ThisRoutine = ComponentData(iComp).ModuleF
    ElseIf iModule = 4 Then
        ThisRoutine = ComponentData(iComp).ModuleB
    End If
    ThisComp = Val(ComponentData(iComp).Label)
    If ThisRoutine = "AV" Then
        Call RoutineAV(ThisComp, ComponentData(iComp).ManualVariability)
        MtxChange(ThisComp) = True
    End If

Next

End Sub
```

```
Public Sub Phase2()
'Calls the Phase 2 routines (DA, DV, DZ, DX, and CV)
'Go through DKType array in order and execute these routines

Dim iDK As Integer
Dim iComp As Integer
Dim ThisComp As Integer
Dim ThisDK As Integer
Dim ThisRoutine As String
Dim booDK As Boolean
Dim DK2 As Integer

For iDK = 1 To MaxDefs
    If iModule = 2 Then
        booDK = DKData(iDK).DKA
    ElseIf iModule = 3 Then
        booDK = DKData(iDK).DKF
    ElseIf iModule = 4 Then
        booDK = DKData(iDK).DKB
    End If

    If booDK = True Then
        DKtoUse = DKData(iDK).DKComp
        If DKData(iDK).DKAddends = True Then
            Call CalcDK(DKtoUse)
            MtxChange(DKtoUse) = True
        End If
        For iComp = 1 To CompsMax
            If iModule = 2 Then
                ThisRoutine = ComponentData(iComp).ModuleA
                ThisDK = ComponentData(iComp).DistribA
                DK2 = ComponentData(iComp).MiscA
            ElseIf iModule = 3 Then
                ThisRoutine = ComponentData(iComp).ModuleF
                ThisDK = ComponentData(iComp).DistribF
                DK2 = ComponentData(iComp).MiscF
            ElseIf iModule = 4 Then
                ThisRoutine = ComponentData(iComp).ModuleB
                ThisDK = ComponentData(iComp).DistribB
                DK2 = ComponentData(iComp).MiscB
            End If
```

```
            ThisComp = Val(ComponentData(iComp).Label)

            If ThisRoutine = "DA" And ThisDK = DKtoUse Then
                Call RoutineDA(ThisComp, ThisDK)
                MtxChange(ThisComp) = True
            ElseIf ThisRoutine = "DV" And ThisDK = DKtoUse Then
                Call RoutineDV(ThisComp, ThisDK)
                MtxChange(ThisComp) = True
            ElseIf ThisRoutine = "DZ" And ThisDK = DKtoUse Then
                Call RoutineDZ(ThisComp, ThisDK, DK2)
                MtxChange(ThisComp) = True
            ElseIf ThisRoutine = "DX" And ThisDK = DKtoUse Then
                Call RoutineDX(ThisComp, ThisDK, DK2)
                MtxChange(ThisComp) = True
            ElseIf ThisRoutine = "CV" And ThisDK = DKtoUse Then
                Call RoutineCV(ThisComp, ThisDK)
                MtxChange(ThisComp) = True
            End If
        Next
    End If
Next


End Sub
```

```
Public Sub Phase3()
'Calls the Phase 3 routines (CP)

'Go through the ComponentData array for the current module, and execute any CP routines
Dim iComp As Integer
Dim ThisComp As Integer
Dim Distrib As Integer
Dim Misc As Integer
Dim ThisRoutine As String

For iComp = 1 To CompsMax
    If iModule = 1 Then
        ThisRoutine = ComponentData(iComp).Module1
        Distrib = ComponentData(iComp).Distrib1
        Misc = ComponentData(iComp).Misc1
    ElseIf iModule = 2 Then
        ThisRoutine = ComponentData(iComp).ModuleA
        Distrib = ComponentData(iComp).DistribA
        Misc = ComponentData(iComp).MiscA
    ElseIf iModule = 3 Then
        ThisRoutine = ComponentData(iComp).ModuleF
        Distrib = ComponentData(iComp).DistribF
        Misc = ComponentData(iComp).MiscF
    ElseIf iModule = 4 Then
        ThisRoutine = ComponentData(iComp).ModuleB
        Distrib = ComponentData(iComp).DistribB
        Misc = ComponentData(iComp).MiscB
    End If

    ThisComp = Val(ComponentData(iComp).Label)

    If ThisRoutine = "CP" Then
        Call RoutineCP(ThisComp, Distrib, Misc)
        MtxChange(ThisComp) = True
        MtxChange(Misc) = True
    End If
Next
```

```
End Sub

Public Sub Phase4()
'Calls the Phase 4 routines (ST, SA, and AJ)
Dim iComp As Integer
Dim ThisComp As Integer
Dim Distrib As Integer
Dim Misc As Integer

For iComp = 1 To CompsMax
    If iModule = 1 Then
        ThisRoutine = ComponentData(iComp).Module1
        Distrib = ComponentData(iComp).Distrib1
        Misc = ComponentData(iComp).Misc1
    ElseIf iModule = 2 Then
        ThisRoutine = ComponentData(iComp).ModuleA
        Distrib = ComponentData(iComp).DistribA
        Misc = ComponentData(iComp).MiscA
    ElseIf iModule = 3 Then
        ThisRoutine = ComponentData(iComp).ModuleF
        Distrib = ComponentData(iComp).DistribF
        Misc = ComponentData(iComp).MiscF
    ElseIf iModule = 4 Then
        ThisRoutine = ComponentData(iComp).ModuleB
        Distrib = ComponentData(iComp).DistribB
        Misc = ComponentData(iComp).MiscB
    End If

    ThisComp = Val(ComponentData(iComp).Label)
    If ThisRoutine = "ST" Then
        Call RoutineST(ThisComp, Distrib)
        MtxChange(ThisComp) = True
    ElseIf ThisRoutine = "SA" Then
        Call RoutineSA(ThisComp, Distrib)
        MtxChange(ThisComp) = True
    ElseIf ThisRoutine = "AJ" Then
        Call RoutineAJ(ThisComp, Distrib, Misc)
        MtxChange(ThisComp) = True
    End If
Next

End Sub
```

```
Public Sub RoutineAV(SC, VF)
'Use Variability Factor (VF) to Calculate Non-Variable Portion of a Component (SC)
'i.e.  (1 - factor) / factor    *  total variable (class 198)   = non-variable portion (class 199)
Dim SCSheet As String
Dim SCCol As Integer

'Get worksheet and column to write formulas onto
SCSheet = CompSheet(SC)
SCCol = CompColumn(SC)
Sheets(SCSheet).Activate
'Write formulas into non-subtotal classes

For iClass = 1 To ClassMax
    ThisClass = MailClassData(iClass).Number
    If ThisClass > 0 And ThisClass < OtherClass And MailClassData(iClass).SubTotal = False Then
        Cells(ClassRow(ThisClass), SCCol).Select
        LineFormula = "=InputMatrix!R" & SC + MatrixRowOffset & "C" & ThisClass + MatrixColOffset
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True Then
        Cells(ClassRow(ThisClass), SCCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    ElseIf ThisClass = OtherClass Then
        Cells(ClassRow(OtherClass), SCCol).Select
        LineFormula = "= (1 - " & VF & ") / " & VF & " * R[-1]C" & SCCol
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    End If
Next

End Sub
```

```
Public Sub RoutineDA(SC, DK)
'Distribute Total in subject component (SC) based on all non-total classes of distribution key (DK).

Dim SCCol As Integer, DKCol As Integer
Dim SCSheet As String, DKSheet As String
Dim TotRow As Integer
Dim ThisClass As Integer
Dim iClass As Integer
Dim LineFormula As String

'Get the physical locations of the subject component and distribution key component
SCCol = CompColumn(SC)
SCSheet = CompSheet(SC)
DKCol = CompColumn(DK)
DKSheet = CompSheet(DK)

Sheets(SCSheet).Activate
'Write formulas
'Write total cost into appropriate cell
LineFormula = "=InputMatrix!R" & SC + MatrixRowOffset & "C" & TotClass + MatrixColOffset
Cells(ClassRow(TotClass), SCCol).Select
Selection.Formula = LineFormula
Selection.NumberFormat = "#,##0"

For iClass = 1 To ClassMax
    'Be sure not to overwrite the total class amount in subject component
    ThisClass = MailClassData(iClass).Number
    If ThisClass > 0 And ThisClass <> TotClass And MailClassData(iClass).SubTotal = False Then
        Cells(ClassRow(ThisClass), SCCol).Select
        TotRow = ClassRow(TotClass)
        LineFormula = "=if(" & DKSheet & "!R" & TotRow & "C" & DKCol & _
            "<> 0, R" & TotRow & "C" & SCCol & " / " & DKSheet & "!R" & TotRow & "C" & DKCol & _
            " * " & DKSheet & "!R" & ClassRow(ThisClass) & "C" & DKCol & ", 0)"
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True And ThisClass <> TotClass Then
        Cells(ClassRow(ThisClass), SCCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    End If
Next
End Sub
```

```
Public Sub RoutineDV(SC, DK)
'Distribute Total in subject component (SC) based on all non-total classes of distribution key (DK)
'(excluding class 199)
'ie. Component 100% variable

Dim SCCol As Integer, DKCol As Integer
Dim SCSheet As String, DKSheet As String
Dim TotRow As Integer, TotVVRow As Integer
Dim ThisClass As Integer
Dim iClass As Integer
Dim LineFormula As String

'Get the physical locations of the subject component and distribution key component
SCCol = CompColumn(SC)
SCSheet = CompSheet(SC)
DKCol = CompColumn(DK)
DKSheet = CompSheet(DK)

Sheets(SCSheet).Activate
'Write formulas

'Write total cost into appropriate cell
LineFormula = "=InputMatrix!R" & SC + MatrixRowOffset & "C" & TotClass + MatrixColOffset
Cells(ClassRow(TotClass), SCCol).Select
Selection.Formula = LineFormula
Selection.NumberFormat = "#,##0"

For iClass = 1 To ClassMax
    'Be sure not to overwrite the total class amount in subject component
    ThisClass = MailClassData(iClass).Number
    If ThisClass > 0 And ThisClass <= TotVVClass And MailClassData(iClass).SubTotal = False Then
        Cells(ClassRow(ThisClass), SCCol).Select
        TotRow = ClassRow(TotClass)
        TotVVRow = ClassRow(TotVVClass)
        LineFormula = "=R" & TotRow & "C" & SCCol & " / " & DKSheet & "!R" & TotVVRow & "C" & DKCol & _
            " * " & DKSheet & "!R" & ClassRow(ThisClass) & "C" & DKCol
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf ThisClass = OtherClass Then
        Cells(ClassRow(OtherClass), SCCol).Select
        Selection.Value = 0
```

```
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True And ThisClass <> TotClass Then
        Cells(ClassRow(ThisClass), SCCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    End If
Next

End Sub
```

```
Public Sub RoutineDZ(SC, DK1, DK2)
'Distribute costs in subject component (SC) based on amounts in all
'non-total classes of the distribution key (DK1)
'(including class 199) with amount  to be fully distributed determined by another component (DK2)
'  Move non variable portion of component to a holding area.
'  Distribute costs based on Distribution Key
'  Add non-variable portion back to class 199

Dim SCCol As Integer, DK1Col As Integer, DK2Col As Integer
Dim SCSheeta As String, DK1Sheet As String, DK2Sheet As String
Dim TotRow As Integer
Dim ThisClass As Integer
Dim iClass As Integer
Dim LineFormula As String
Dim HoldCost As Double

'Get the physical locations of the subject component and distribution key components
SCCol = CompColumn(SC)
SCSheet = CompSheet(SC)
DK1Col = CompColumn(DK1)
DK1Sheet = CompSheet(DK1)
DK2Col = CompColumn(DK2)
DK2Sheet = CompSheet(DK2)


Sheets(SCSheet).Activate

'Write total cost into appropriate cell
LineFormula = "=InputMatrix!R" & SC + MatrixRowOffset & "C" & TotClass + MatrixColOffset
Cells(ClassRow(TotClass), SCCol).Select
Selection.Formula = LineFormula
Selection.NumberFormat = "#,##0"

'First find the portion of costs to set aside using DK2
'Write this underneath the TotClass
LineFormula = "=" & DK2Sheet & "!R" & ClassRow(OtherClass) & "C" & DK2Col & " / " & _
              DK2Sheet & "!R" & ClassRow(TotClass) & "C" & DK2Col & " * InputMatrix!R" & SC + MatrixRowOffset & _
              "C" & TotClass + MatrixColOffset
TotRow = ClassRow(TotClass)
Cells(TotRow + 1, SCCol).Select
Selection.Formula = LineFormula
Selection.NumberFormat = "#,##0"
```

```
'Get the total for the subject component minus the costs to set aside and write that underneath
LineFormula = "=InputMatrix!R" & SC + MatrixRowOffset & "C" & TotClass + MatrixColOffset & " - R[-1]C"
Cells(TotRow + 2, SCCol).Select
Selection.Formula = LineFormula
Selection.NumberFormat = "#,##0"

'Distribute the subject component total costs MINUS THE AMOUNT SET ASIDE USING DK2 (IN TotRow + 2)
For iClass = 1 To ClassMax
    'Be sure not to overwrite the total class amount in subject component
    ThisClass = MailClassData(iClass).Number
    If ThisClass > 0 And ThisClass <= TotVVClass And MailClassData(iClass).SubTotal = False Then
        Cells(ClassRow(ThisClass), SCCol).Select
        LineFormula = "=R" & TotRow + 2 & "C" & SCCol & " / " & DK1Sheet & "!R" & TotRow & "C" & DK1Col & _
            " * " & DK1Sheet & "!R" & ClassRow(ThisClass) & "C" & DK1Col
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf ThisClass = OtherClass Then
        'Get distribution and then add back in amount set aside
        Cells(ClassRow(OtherClass), SCCol).Select
        LineFormula = "=(R" & TotRow + 2 & "C" & SCCol & " / " & DK1Sheet & "!R" & TotRow & "C" & DK1Col & _
            " * " & DK1Sheet & "!R" & ClassRow(ThisClass) & "C" & DK1Col & ") + R" & TotRow + 1 & "C" & SCCol
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True And ThisClass <= TotVVClass Then
        Cells(ClassRow(ThisClass), SCCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    End If
Next

End Sub
```

```
Public Sub RoutineDX(SC, DK1, DK2)
'Distribute costs in subject component (SC) based on amounts in all
'non-total classes of the distribution key (DK1)
'(excluding class 199) with variability determined by another component (DK2)

Dim SCCol As Integer, DK1Col As Integer, DK2Col As Integer
Dim SCSheet As String, DK1Sheet As String, DK2Sheet As String
Dim TotVVRow As Integer
Dim ThisClass As Integer
Dim iClass As Integer
Dim LineFormula As String
Dim HoldCost As Double

'Get the physical locations of the subject component and distribution key components
SCCol = CompColumn(SC)
SCSheet = CompSheet(SC)
DK1Col = CompColumn(DK1)
DK1Sheet = CompSheet(DK1)
DK2Col = CompColumn(DK2)
DK2Sheet = CompSheet(DK2)

Sheets(SCSheet).Activate
'Write formulas

'Write total cost into appropriate cell
LineFormula = "=InputMatrix!R" & SC + MatrixRowOffset & "C" & TotClass + MatrixColOffset

'Calculate the institutional costs using DK2
LineFormula = "=" & DK2Sheet & "!R" & ClassRow(OtherClass) & "C" & DK2Col & " / " & _
              DK2Sheet & "!R" & ClassRow(TotClass) & "C" & DK2Col & " * InputMatrix!R" & SC + MatrixRowOffset & _
              "C" & TotClass + MatrixColOffset
Cells(ClassRow(OtherClass), SCCol).Select
Selection.Formula = LineFormula
Selection.NumberFormat = "#,##0"

'Calculate total volume variable costs in SC by subtracting institutional costs from total costs
LineFormula = "=R[+2]C - R[+1]C"
Cells(ClassRow(TotVVClass), SCCol).Select
Selection.Formula = LineFormula
Selection.NumberFormat = "#,##0"

TotVVRow = ClassRow(TotVVClass)
```

```
'Distribute the subject component volume variable costs using DK1
For iClass = 1 To ClassMax
    'Be sure not to overwrite the total class amount in subject component
    ThisClass = MailClassData(iClass).Number
    If ThisClass > 0 And ThisClass < TotVVClass And MailClassData(iClass).SubTotal = False Then
        Cells(ClassRow(ThisClass), SCCol).Select
        LineFormula = "=R" & TotVVRow & "C" & SCCol & " / " & DK1Sheet & "!R" & TotVVRow & "C" & DK1Col & _
            " * " & DK1Sheet & "!R" & ClassRow(ThisClass) & "C" & DK1Col
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True And ThisClass < TotVVClass Then
        Cells(ClassRow(ThisClass), SCCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    End If
Next

End Sub
```

```
Public Sub RoutineCV(SC, DK)
'Compute non-variable portion (class 199) of a component (SC) based on a second component (DK)

Dim SCCol As Integer, DKCol As Integer
Dim SCSheet As String, DKSheet As String
Dim ThisClass As Integer
Dim iClass As Integer
Dim LineFormula As String

'Get the physical locations of the subject component and distribution key component
SCCol = CompColumn(SC)
SCSheet = CompSheet(SC)
DKCol = CompColumn(DK)
DKSheet = CompSheet(DK)

Sheets(SCSheet).Activate
'Bring in volume variable costs from InputMatrix

For iClass = 1 To ClassMax
    ThisClass = MailClassData(iClass).Number
    If ThisClass > 0 And ThisClass <> OtherClass And MailClassData(iClass).SubTotal = False Then
        Cells(ClassRow(ThisClass), SCCol).Select
        LineFormula = "=InputMatrix!R" & SC + MatrixRowOffset & "C" & ThisClass + MatrixColOffset
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True Then
        Cells(ClassRow(ThisClass), SCCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    ElseIf ThisClass = OtherClass Then
        LineFormula = "=" & DKSheet & "!R" & ClassRow(OtherClass) & "C" & DKCol & _
                      "/" & DKSheet & "!R" & ClassRow(TotVVClass) & "C" & DKCol & " * R[-1]C"
        Cells(ClassRow(OtherClass), SCCol).Select
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    End If
Next

End Sub
```

```
Public Sub RoutineCP(SC, PC, PP)
'Calculate PESSA Costs.
'  Subtract the variable classes of one component (PP) (Pre-Pessa) from the comprable classses of a
'  second PESSA component (PC) and store results in the subject component (SC).
'  Set the non-variable portion of the Pre PESSA component equal to the original  total minus the
'  sum of variable  classes in the recipient component.

Dim SCCol As Integer, PCCol As Integer, PPCol As Integer
Dim SCSheet As String, PCSheet As String, PPSheet As String
Dim ThisRow As Integer
Dim ThisClass As Integer
Dim iClass As Integer
Dim LineFormula As String

'Get the physical locations of the subject component and distribution key components
SCCol = CompColumn(SC)
SCSheet = CompSheet(SC)
PCCol = CompColumn(PC)
PCSheet = CompSheet(PC)
PPCol = CompColumn(PP)  'Get this one from InputMatrix!!!!!
PPSheet = CompSheet(PP)

Sheets(SCSheet).Activate
'Write formulas onto first component, which is the difference between the component after PESSA
'distributions and before PESSA distributions

For iClass = 1 To ClassMax
    ThisClass = MailClassData(iClass).Number
    ThisRow = ClassRow(ThisClass)
    If ThisClass > 0 And ThisClass <= TotVVClass And MailClassData(iClass).SubTotal = False Then
        Cells(ClassRow(ThisClass), SCCol).Select
        LineFormula = "=" & PCSheet & "!R" & ThisRow & "C" & PCCol & "-" & PPSheet & "!R" & _
                      ThisRow & "C" & PPCol
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True Then
        Cells(ClassRow(ThisClass), SCCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    ElseIf ThisClass = OtherClass Then
        Cells(ClassRow(OtherClass), SCCol).Select
        Selection.Value = 0
```

```
        Selection.NumberFormat = "#,##0"
    End If
Next
'The Other Cost for the Pre-PESSA component is set equal to the Other Cost for the PESSA component,
'and the Pre-PESSA Total Cost class is recalculated.

Sheets(PPSheet).Activate
Cells(ClassRow(OtherClass), PPCol).Select
LineFormula = "=" & PCSheet & "!R" & ClassRow(OtherClass) & "C" & PCCol
Selection.Formula = LineFormula
Selection.NumberFormat = "#,##0"

Cells(ClassRow(TotClass), PPCol).Select
LineFormula = ClassFormulas(TotClass)
Selection.Formula = LineFormula
Selection.NumberFormat = "#,##0"

End Sub
```

```
Public Sub RoutineST(SC, AC)
'Set component (SC) total (class 200) equal to the total (class 200) of another component (OC).

Dim SCCol, ACCol As Integer
Dim SCSheet, ACSheet As String
Dim LineFormula As String

'Get the physical locations of the subject component and distribution key component
SCCol = CompColumn(SC)
SCSheet = CompSheet(SC)
ACCol = CompColumn(AC)
ACSheet = CompSheet(AC)

'The Other Cost for the subject component is set equal to the Total Cost of the other component,
'and the subject component Total Cost class is recalculated.

Sheets(SCSheet).Activate
Cells(ClassRow(OtherClass), SCCol).Select
LineFormula = "=" & ACSheet & "!R" & ClassRow(TotClass) & "C" & ACCol
Selection.Formula = LineFormula
Selection.NumberFormat = "#,##0"

Cells(ClassRow(TotClass), SCCol).Select
LineFormula = ClassFormulas(TotClass)
Selection.Formula = LineFormula
Selection.NumberFormat = "#,##0"

End Sub
```

```
Public Sub RoutineSA(SC, AC)
'Set all values in a component (SC) equal to the values in another component (AC).

Dim SCCol As Integer, ACCol As Integer
Dim SCSheet As String, ACSheet As String
Dim ThisRow As Integer
Dim ThisClass As Integer
Dim iClass As Integer
Dim LineFormula As String

'Get the physical locations of the subject component and distribution key component
SCCol = CompColumn(SC)
SCSheet = CompSheet(SC)
ACCol = CompColumn(AC)
ACSheet = CompSheet(AC)

Sheets(SCSheet).Activate

For iClass = 1 To ClassMax
    ThisClass = MailClassData(iClass).Number
    ThisRow = ClassRow(ThisClass)
    If ThisClass > 0 And MailClassData(iClass).SubTotal = False Then
        Cells(ClassRow(ThisClass), SCCol).Select
        LineFormula = "=" & ACSheet & "!R" & ThisRow & "C" & ACCol
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True Then
        Cells(ClassRow(ThisClass), SCCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    End If
Next
End Sub
```

```
Public Sub RoutineAJ(SC, FC, AC)
'Multiply Subject Component (SC) by Factor Component (FC), Add results to subject component
'and subtract results from adjustment class (AC)
'Add a column to the WorkArea sheet to hold the results of multiplying subject component by factor
'component.  The sum of this amount is then subtracted from the adjustment class in the subject
'component

Dim SCCol As Integer, FCCol As Integer
Dim SCSheet As String, FCSheet As String
Dim ThisClass As Integer
Dim iClass As Integer
Dim LineFormula As String

'Get the physical locations of the subject component and factor component
SCCol = CompColumn(SC)
SCSheet = CompSheet(SC)
FCCol = CompColumn(FC)
FCSheet = CompSheet(FC)

Sheets("WorkArea").Activate
'WorkAreaCol holds the next free column on this worksheet
If WorkAreaCol < FirstColumn Then
    WorkAreaCol = FirstColumn
    'This is the first adjustment column added to this worksheet
End If

'Make a title for this column
Cells(CompNameRow, WorkAreaCol).Select
LineFormula = "=" & SCSheet & "!R" & CompNameRow & "C" & SCCol
Selection.Formula = LineFormula
Selection.WrapText = True
Selection.HorizontalAlignment = xlCenter
Columns(WorkAreaCol).ColumnWidth = 15


'Multiply subject component by factor component and put this in work area
For iClass = 1 To ClassMax
    ThisClass = MailClassData(iClass).Number
    ThisRow = ClassRow(ThisClass)
    If ThisClass > 0 And MailClassData(iClass).SubTotal = False Then
        Cells(ClassRow(ThisClass), WorkAreaCol).Select
        LineFormula = "=InputMatrix!R" & SC + MatrixRowOffset & "C" & ThisClass + MatrixColOffset & _
```

```
                        " * " & FCSheet & "!R" & ThisRow & "C" & FCCol
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True Then
        Cells(ClassRow(ThisClass), WorkAreaCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    End If
Next

'Subtract adjustment costs in workarea from InputMatrix costs for component
Sheets(SCSheet).Activate
For iClass = 1 To ClassMax
    ThisClass = MailClassData(iClass).Number
    ThisRow = ClassRow(ThisClass)
    If ThisClass > 0 And MailClassData(iClass).SubTotal = False And ThisClass <> AC Then
        Cells(ClassRow(ThisClass), SCCol).Select
        LineFormula = "=InputMatrix!R" & SC + MatrixRowOffset & "C" & ThisClass + MatrixColOffset & _
                      " + WorkArea!R" & ThisRow & "C" & WorkAreaCol
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True Then
        Cells(ClassRow(ThisClass), SCCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    ElseIf ThisClass = AC Then
        Cells(ClassRow(AC), SCCol).Select
        LineFormula = "=InputMatrix!R" & SC + MatrixRowOffset & "C" & AC + MatrixColOffset & _
                      " - WorkArea!R" & ClassRow(TotVVClass) & "C" & WorkAreaCol
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    End If
Next

'Set work area pointer to next column
WorkAreaCol = WorkAreaCol + 1


End Sub
```

```
Sub CalcDK(DK)
'Calculates the distribution key if the number of addends in the key is greater than 1

Dim iDetail As Integer, iClass As Integer, iAddends As Integer
Dim LineFormula As String
Dim DKCol As Integer
Dim DKSheet As String
Dim CurCol As Integer, LastCol As Integer, FirstCol As Integer
Dim CurSheet As String, LastSheet As String
Dim booSeq As Boolean, booSheet As Boolean
Dim MaxAddends As Integer

'Get the worksheet and column on worksheet to write the formula
DKCol = CompColumn(DK)
DKSheet = CompSheet(DK)

'Find the addend list in the DKDetail array

For iDetail = 1 To DetailMax
    If DKDetail(iDetail).DKNumber = DK Then
        Exit For
    End If
Next

'Initialize the strings holding the formula.  The strings will contain a
'sum formula for all the components(addends) comprising the DK

'Go down the addend list, and find components on their associated worksheets.  If in
'column order on the worksheet then we will use shorthand notation

MaxAddends = DKDetail(iDetail).AddendCount
LineFormula = "=Sum("
For iAddends = 1 To MaxAddends
    ThisAddend = DKDetail(iDetail).Addends(iAddends)
    CurSheet = CompSheet(ThisAddend)
    CurCol = CompColumn(ThisAddend)

        'This booSeq is used to determine if the DK addends are in column
        'order on the various cost segment worksheet.  If booSeq = True then the DK list follows the
        'same order as the columns on the current worksheet.  This is useful because the
        'SUM formula allows you to identify a range of cells.  For example, we can
        'write SUM(RC5:RC10), which will sum columns 5 through 10.
```

```
    'LastCol holds the last column looked at on the current worksheet.
    'FirstCol holds the first column in the range of columns that are to be summed
    'in order.  In the example above, FirstCol = 5.


If CurCol = LastCol + 1 And CurSheet = LastSheet And booSeq = False Then
    FirstCol = LastCol
    booSeq = True
    booSheet = True
ElseIf CurCol = LastCol + 1 And CurSheet = LastSheet And booSeq = True Then
    booSeq = True
    booSheet = True
ElseIf CurCol <> LastCol + 1 And CurSheet = LastSheet Then
    booSeq = False
    booSheet = True
ElseIf CurSheet <> LastSheet Then
    booSeq = False
    booSheet = False
    'LastCol = 0
End If

'If booSeq is True then we are in a sequence of addends that are in order on a cost
'segment worksheet.  Keep going until we find the last addend in order on the
'cost segment worksheet
'We are always one addend component behind when building the formula string
If booSeq = True And booSheet = True And iAddends < MaxAddends Then
    LastCol = CurCol
    LastSheet = CurSheet
'At the end of a range of columns in order, so add the range to the
'formula string.  Still on the same sheet
ElseIf booSeq = False And booSheet = True And iAddends < MaxAddends Then

    'LastCol = 0 for the first addend only
    If LastCol = 0 Then
        LastCol = CurCol
        LastSheet = CurSheet
        FirstCol = 0

    'The range is added to the formula string
    ElseIf LastCol > 0 And FirstCol > 0 Then
        LineFormula = LineFormula & CurSheet & "!RC" & FirstCol & ":" & CurSheet & "!RC" & LastCol & ","
        LastCol = CurCol
```

```
        LastSheet = CurSheet
        FirstCol = 0

    'The component is in the DK, but is the column is not part of a range of
    'columns.  Add this single column to the formula string.
    ElseIf LastCol > 0 And FirstCol = 0 Then
        LineFormula = LineFormula & CurSheet & "!RC" & LastCol & ","
        LastCol = CurCol
        LastSheet = CurSheet
        FirstCol = 0
    End If

ElseIf booSeq = False And booSheet = False And iAddends < MaxAddends Then
'Going to a new sheet, so finish up the last sheet.

    'LastCol = 0 for the first addend only
    If LastCol = 0 Then
        LastCol = CurCol
        LastSheet = CurSheet
        FirstCol = 0

    'Add the range to the formula string from the prior sheet.
    ElseIf LastCol > 0 And FirstCol > 0 Then
        LineFormula = LineFormula & LastSheet & "!RC" & FirstCol & ":" & LastSheet & "!RC" & LastCol & ","
        LastCol = CurCol
        LastSheet = CurSheet
        FirstCol = 0

    'Add the single component to the formula string from the prior sheet.
    ElseIf LastCol > 0 And FirstCol = 0 Then
        LineFormula = LineFormula & LastSheet & "!RC" & LastCol & ","
        LastCol = CurCol
        LastSheet = CurSheet
        FirstCol = 0
    End If

'The last component (addend) of this distribution key, so finish off the string
'Have to add the prior column and the current column this time.
ElseIf booSeq = False And booSheet = True And iAddends = MaxAddends Then
'The last addend is on the same worksheet

    'This column is not part of a range of columns, add it separatly.
```

```
        If FirstCol = 0 Then
            LineFormula = LineFormula & CurSheet & "!RC" & LastCol & "," & CurSheet & "!RC" & CurCol & ")"

        'The end of a range of columns to add to the formula string
        ElseIf FirstCol > 0 Then
            LineFormula = LineFormula & CurSheet & "!RC" & FirstCol & ":" & CurSheet & "!RC" & LastCol & "," & CurSheet
& "!RC" & CurCol & ")"
        End If

    ElseIf booSeq = False And booSheet = False And iAddends = MaxAddends Then
    'The last addend is on a new worksheet

        'This column is not part of a range of columns, add it separatly.
        If FirstCol = 0 Then
            LineFormula = LineFormula & LastSheet & "!RC" & LastCol & "," & CurSheet & "!RC" & CurCol & ")"

        'The end of a range of columns to add to the formula string
        ElseIf FirstCol > 0 Then
            LineFormula = LineFormula & LastSheet & "!RC" & FirstCol & ":" & LastSheet & "!RC" & LastCol & "," &
CurSheet & "!RC" & CurCol & ")"
        End If

    'We are still in a range of columns, obviously on the same sheet, so go from FirstC1 to the last column
    ElseIf booSeq = True And iAddends = MaxAddends Then
        LineFormula = LineFormula & CurSheet & "!RC" & FirstCol & ":" & CurSheet & "!RC" & CurCol & ")"
    End If

Next iAddends

'Now write the formula into the distribution key component

Sheets(DKSheet).Activate

For iClass = 1 To ClassMax
    'Assume subtotal classes already
    ThisClass = MailClassData(iClass).Number
    If ThisClass > 0 And MailClassData(iClass).SubTotal = False Then
        Cells(ClassRow(ThisClass), DKCol).Select
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True Then
        Cells(ClassRow(ThisClass), DKCol).Select
```

```
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    End If
Next
Calculate

End Sub
```

```
Sub RunDRoutines()
Dim I As Integer

'Initialize Component Change Matrix to False

For I = 1 To DMax
    If DData(I).DRoutine = "AA" Then
        If DData(I).DParm3 = OtherClass Then
            Call RoutineAA(DData(I).DComp, DData(I).DParm1, DData(I).DParm2, DData(I).DParm3)
            MtxChange(DData(I).DComp) = True
        Else
            Call RoutineAA(DData(I).DComp, DData(I).DParm1, DData(I).DParm2, 0)
            MtxChange(DData(I).DComp) = True
        End If
    ElseIf DData(I).DRoutine = "AD" Then
        If DData(I).DParm3 > 0 Then
            Call RoutineAD(DData(I).DComp, DData(I).DParm1, DData(I).DParm2, DData(I).DParm3)
            MtxChange(DData(I).DComp) = True
        Else
            Call RoutineAD(DData(I).DComp, DData(I).DParm1, DData(I).DParm2, 0)
            MtxChange(DData(I).DComp) = True
        End If
    ElseIf DData(I).DRoutine = "PS" Then
        Call RoutinePS(DData(I).DComp, DData(I).DParm1)
        MtxChange(DData(I).DComp) = True
    End If
    Calculate
Next

End Sub
```

```
Public Sub RoutineAA(NC, SC, FC, AC)
'Multiply Subject Component (SC) by Factor Component (FC), Add results to subject component
'and subtract results from adjustment class (AC).  Place results in new component (NC).

'Note:  If AC = 0 Then do not subtract results from adjustment class (i.e. component cost grows)
'       AC will always be either 0 or 199

Dim SCCol As Integer, FCCol As Integer
Dim SCSheet As String, FCSheet As String
Dim ThisClass As Integer
Dim iClass As Integer
Dim LineFormula As String

'Get the physical locations of the subject component, factor component, and new component
SCCol = CompColumn(SC)
SCSheet = CompSheet(SC)
FCCol = CompColumn(FC)
FCSheet = CompSheet(FC)
NCCol = CompColumn(NC)
NCSheet = CompSheet(NC)

Sheets(NCSheet).Activate

'Multiply subject component by factor component and put this in new component
For iClass = 1 To ClassMax
    ThisClass = MailClassData(iClass).Number
    ThisRow = ClassRow(ThisClass)
    If ThisClass > 0 And MailClassData(iClass).SubTotal = False Then
        Cells(ThisRow, NCCol).Select
        LineFormula = "=" & SCSheet & "!R" & ThisRow & "C" & SCCol & _
                      " * " & FCSheet & "!R" & ThisRow & "C" & FCCol
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True Then
        Cells(ClassRow(ThisClass), NCCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    End If
Next

'Subtract adjustment costs in workarea from InputMatrix costs for component
If AC = OtherClass Then
```

```
    Cells(ClassRow(AC), NCCol).Select
    LineFormula = "=-R[-1]C"
    Selection.NumberFormat = "#,##0"
    Selection.Formula = LineFormula
End If

End Sub
```

```
Public Sub RoutineAD(SC, A1, A2, A3)
'Add Components A1, A2, and A3 (if any) and place results in subject component (SC)

Dim SCCol As Integer
Dim A1Col As Integer
Dim A2Col As Integer
Dim A3Col As Integer
Dim SCSheet As String
Dim A1Sheet As String
Dim A2Sheet As String
Dim A3Sheet As String

Dim ThisClass As Integer
Dim iClass As Integer
Dim LineFormula As String

'Get the physical locations of the subject component, and addend components
SCCol = CompColumn(SC)
SCSheet = CompSheet(SC)
A1Col = CompColumn(A1)
A1Sheet = CompSheet(A1)
A2Col = CompColumn(A2)
A2Sheet = CompSheet(A2)
If A3 > 0 Then
    A3Col = CompColumn(A3)
    A3Sheet = CompSheet(A3)
End If

Sheets(SCSheet).Activate

'Add together A1, A2, and A3 and put in subject component
For iClass = 1 To ClassMax
    ThisClass = MailClassData(iClass).Number
    ThisRow = ClassRow(ThisClass)
    If ThisClass > 0 And MailClassData(iClass).SubTotal = False Then
        Cells(ThisRow, SCCol).Select
        LineFormula = "=" & A1Sheet & "!R" & ThisRow & "C" & A1Col & _
                      " + " & A2Sheet & "!R" & ThisRow & "C" & A2Col
        If A3 > 0 Then
            LineFormula = LineFormula & " + " & A3Sheet & "!R" & ThisRow & "C" & A3Col
        End If
        Selection.Formula = LineFormula
```

```
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True Then
        Cells(ClassRow(ThisClass), SCCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    End If
Next

'Set adjustment class = sum of total volume variable * -1
If AC = OtherClass Then
    Cells(ClassRow(AC), NCCol).Select
    LineFormula = "=-R[-1]C"
    Selection.NumberFormat = "#,##0"
    Selection.Formula = LineFormula
End If

End Sub
```

```
Public Sub RoutinePS(SC, AC)
'Adjusts Subject Component (SC) by doing the following:
'   Set Adjustment Class (199) to 0
'   Sum up all non-zero classes and put that in AC
'   Reverse signs of all volume variables

Dim SCCol As Integer
Dim SCSheet As String

Dim ThisClass As Integer
Dim iClass As Integer
Dim LineFormula As String

'Get the physical locations of the subject component
SCCol = CompColumn(SC)
SCSheet = CompSheet(SC)

Sheets(SCSheet).Activate

'First reverse signs

'Multiply attributable classes by -1
For iClass = 1 To ClassMax
    ThisClass = MailClassData(iClass).Number
    ThisRow = ClassRow(ThisClass)
    If ThisClass > 0 And MailClassData(iClass).SubTotal = False And ThisClass <> OtherClass Then
        Cells(ThisRow, SCCol).Select
        LineFormula = Selection.Formula
        LineFormula = LineFormula & " * -1"
        Selection.Formula = LineFormula
        Selection.NumberFormat = "#,##0"
    ElseIf MailClassData(iClass).SubTotal = True Then
        Cells(ClassRow(ThisClass), SCCol).Select
        Selection.Formula = ClassFormulas(ThisClass)
        Selection.NumberFormat = "#,##0"
    End If
Next

'Set adjustment class (199) = sum of total volume variable * -1
If AC = OtherClass Then
    Cells(ClassRow(AC), SCCol).Select
    LineFormula = "=-R[-1]C"
```

```
    Selection.NumberFormat = "#,##0"
    Selection.Formula = LineFormula
End If


End Sub
```

D. SubTotals – ModSubtotals

```
Sub ClassSubtotals()

'I, J, K are looping counters
Dim iClass As Integer, iClass2 As Integer

'LineFOrmula used to build cell formulas
Dim LineFormula As String

'ParentClass is the parent class we generating a formula to subtotal its children
Dim ParentClass As Integer

'ChildClass is a child of the ParentClass
Dim ChildRow, ChildClass As Integer

'FirstSum is True if first child of the parent has not yet been added to formula
Dim FirstSum As Boolean

ReDim ClassFormulas(MaxClass)

'Find each subtotal class
'Build the formula to do that subtotal
For iClass = 1 To ClassMax
    If MailClassData(iClass).SubTotal = True Then
    ParentClass = MailClassData(iClass).Number
    'Build the formula for this Parent class
    'by finding all of the classes' children
        LineFormula = "=SUM("
        FirstSum = True
        For iClass2 = 1 To ClassMax
            If MailClassData(iClass2).Parent = ParentClass Then
                ChildClass = MailClassData(iClass2).Number
                ChildRow = ClassRow(ChildClass)
                If FirstSum = True Then
                    LineFormula = LineFormula & "R" & ChildRow & "C"
                    FirstSum = False
                Else
                    LineFormula = LineFormula & ",R" & ChildRow & "C"
                End If
            End If
```

```
            If iClass2 = ClassMax Then
                LineFormula = LineFormula & ")"
            End If

        'Get the next child class for this parent class
        Next
        ClassFormulas(ParentClass) = LineFormula

    End If

'Get the next parent class
Next

End Sub
```

```
'Subroutine ComponentSubtotals puts in the formulas in the subtotal component to
'sum up its child components

Sub ComponentSubtotals()

Dim iComp As Integer, iComp2 As Integer
Dim iClass As Integer
Dim booSubtotal As Boolean
Dim ParentComp As Integer
Dim ChildComp As Integer
Dim LineFormula As String
Dim FirstSum As Boolean
Dim SubtotCol As Integer
Dim ThisParent As Integer
'Dim ChildComp As Integer
Dim ChildCol As Integer
Dim ChildSheet As String

'Go through the Components array and find each subtotal component.
'Find the subtotal component on the correct cost segment page.
'Then build a formula to add up the child components and put it
'in the subtotal component column.

Application.StatusBar = ModuleLabel(iModule) & " Workbook: Component Subtotals"
Application.Calculation = xlManual
For iComp = 1 To CompsMax
    If iModule = 2 Then
        booSubtotal = ComponentData(iComp).SubTotalA
    ElseIf iModule = 3 Then
        booSubtotal = ComponentData(iComp).SubTotalF
    ElseIf iModule = 4 Then
        booSubtotal = ComponentData(iComp).SubTotalB
    End If
    If booSubtotal = True And ComponentData(iComp).CS > 0 And ComponentData(iComp).CS <= LastCS Then
        ParentComp = Val(ComponentData(iComp).Label)

        LineFormula = "=SUM("
        FirstSum = True

        'Go through ComponentData array and find each child component (i.e. has this subtotal comp
        'as its parent
        For iComp2 = 1 To CompsMax
```

```
        If iModule = 2 Then
            ThisParent = ComponentData(iComp2).ParentA
        ElseIf iModule = 3 Then
            ThisParent = ComponentData(iComp2).ParentF
        ElseIf iModule = 4 Then
            ThisParent = ComponentData(iComp2).ParentB
        ElseIf iModule = 5 Then
            ThisParent = ComponentData(iComp2).ParentC
        End If
        If ThisParent = ParentComp Then
            ChildComp = Val(ComponentData(iComp2).Label)
            ChildCol = CompColumn(ChildComp)
            ChildSheet = CompSheet(ChildComp)
            If FirstSum = True Then
                LineFormula = LineFormula & ChildSheet & "!RC" & ChildCol
                FirstSum = False
            Else
                LineFormula = LineFormula & "," & ChildSheet & "!RC" & ChildCol
            End If
        End If
        If iComp2 = CompsMax Then
            LineFormula = LineFormula & ")"
        End If
    Next

    'Now write the formula in the parent component column

    'Go to the worksheet for the this parent component
    Sheets(CompSheet(ParentComp)).Select
    SubtotCol = CompColumn(ParentComp)

    For iClass = 1 To ClassMax
        ThisClass = MailClassData(iClass).Number
        If ThisClass > 0 Then
            Cells(ClassRow(ThisClass), SubtotCol).Select
            Selection.Formula = LineFormula
            Selection.NumberFormat = "#,##0"
            Selection.HorizontalAlignment = xlRight
        End If
    Next

End If
```

```
    MtxChange(ParentComp) = True
    'Do the next subtotal component
Next

'Go through again to put the C Report subtotals onto the B workbook.
If iModule = 4 Then
  For iComp = 1 To CompsMax
    booSubtotal = ComponentData(iComp).SubTotalC
    If booSubtotal = True And ComponentData(iComp).CS > 0 And ComponentData(iComp).CS <= 20 Then
        ParentComp = Val(ComponentData(iComp).Label)

        LineFormula = "=SUM("
        FirstSum = True

        'Go through ComponentData array and find each child component (i.e. has this subtotal comp
        'as its parent
        For iComp2 = 1 To CompsMax
            ThisParent = ComponentData(iComp2).ParentC
            If ThisParent = ParentComp Then
                ChildComp = Val(ComponentData(iComp2).Label)
                ChildCol = CompColumn(ChildComp)
                ChildSheet = CompSheet(ChildComp)
                If FirstSum = True Then
                    LineFormula = LineFormula & ChildSheet & "!RC" & ChildCol
                    FirstSum = False
                Else
                    LineFormula = LineFormula & "," & ChildSheet & "!RC" & ChildCol
                End If
            End If
            If iComp2 = CompsMax Then
                LineFormula = LineFormula & ")"
            End If
        Next

        'Now write the formula in the parent component column

        'Go to the worksheet for the this parent component
        Sheets(CompSheet(ParentComp)).Select
        SubtotCol = CompColumn(ParentComp)

        For iClass = 1 To ClassMax
            ThisClass = MailClassData(iClass).Number
```

```
            If ThisClass > 0 Then
                Cells(ClassRow(ThisClass), SubtotCol).Select
                Selection.Formula = LineFormula
                Selection.NumberFormat = "#,##0"
                Selection.HorizontalAlignment = xlRight
            End If
        Next

    End If
    MtxChange(ParentComp) = True
    'Do the next subtotal component
  Next
End If

Calculate
End Sub
```

E. Reporting – ModReports

```
'*******************************************************************
'*****  Subroutines for running reports
'*******************************************************************
'       Subroutine RptInit() reads in user-defined public variables and
'       sets values for other public variables
'
'       Subroutine Reports() runs the Input, A, F, B, and C Reports
'
'Here are the rules for page breaks and setting headers:
'
'1.  Generate a page break after a subtotal component, even if the subtotal component is not printed.
'    The subtitle (Header4) for the page is the name of the subtotal component
'2.  Generate a page break if there is a header line in the ComponentData array, even if there is no subtotal component
before this
'    The header4 for this page is:
'       a.  If there are only 2 components before, it is the name of the two components
'       b.  If there are more than 2 components before, then it is the cost segment name
'3.  There will be automatic page breaks after each cost segment since they are on separate worksheets.
'4.  Header4 (subtitle) for cost segments with only one component or one subtotal is the cost segment name.
'5.  If needed, we will add information to the control table to indicate No Page Break.
'*******************************************************************

Sub RptInit()

Dim RYear As String
Dim iComma As String
Dim NumComps As Integer

'Read variables that are used to generate reports
Workbooks(CRABook).Activate
Application.GoTo reference:="ReportSheet"
R1 = Selection.Row
Cells(R1 + 1, 2).Select
IHeader1 = Selection.Value
Cells(R1 + 2, 2).Select
IHeader2 = Selection.Value
Cells(R1 + 3, 2).Select
IStartPrintRows = Selection.Value
Cells(R1 + 4, 2).Select
```

```
AHeader1 = Selection.Value
Cells(R1 + 5, 2).Select
AHeader2 = Selection.Value
Cells(R1 + 6, 2).Select
FHeader1 = Selection.Value
Cells(R1 + 7, 2).Select
FHeader2 = Selection.Value
Cells(R1 + 8, 2).Select
BHeader1 = Selection.Value
Cells(R1 + 9, 2).Select
BHeader2 = Selection.Value
Cells(R1 + 10, 2).Select
CHeader1 = Selection.Value
Cells(R1 + 11, 2).Select
CHeader2 = Selection.Value
Cells(R1 + 12, 2).Select
DHeader1 = Selection.Value
Cells(R1 + 13, 2).Select
DHeader2 = Selection.Value
Cells(R1 + 14, 2).Select
KHeader1 = Selection.Value
Cells(R1 + 15, 2).Select
KHeader2 = Selection.Value
Cells(R1 + 16, 2).Select
KHeader3 = Selection.Value
Cells(R1 + 17, 2).Select
KComps = Selection.Value
Cells(R1 + 18, 2).Select
RptCentury = Selection.Value
Cells(R1 + 19, 2).Select
RptStartRow = Selection.Value
Cells(R1 + 20, 2).Select
RptStartCol = Selection.Value
Cells(R1 + 21, 2).Select
RptCompRow = Selection.Value
Cells(R1 + 22, 2).Select
CalcKeysCS = Selection.Value
Cells(R1 + 23, 2).Select
IPrintDecimals = Selection.Value
Cells(R1 + 24, 2).Select
IRevComp = Selection.Value
```

```
'Read USPSorPRC drop down and set variable accordingly
Application.GoTo reference:="USPSorPRC"
USPSorPRC = Selection.Value

'Build string containing report year - for RF support
If Left(ModName, 2) = "BY" Then
    RYear = "Base Year "
Else
    RYear = "Fiscal Year "
End If

'Add year, which will follow BY or FY in model name
If iRunType = 3 Then
    RYear = RateCase & " - " & RYear & RFYear
Else
    RYear = RYear & RptCentury & Mid(ModName, 3, 2)
End If

If iRunType = 3 Then
    If Rfwd = 1 Then
        RYear = RYear & " Before Workyear Mix Adj."
    ElseIf Rfwd = 2 Then
        RYear = RYear & " After Workyear Mix Adj."
    End If

End If

'Need USPS/PRC flag, to add PRC Version to header is applicable
If USPSorPRC = 2 Then
    RYear = RYear & " (PRC)"
End If

If VerType = 1 And iRunType = 3 Then
    RYear = RYear & " - CONFIDENTIAL"
End If

'Update headers with RYear variable
If iRunType = 3 Then
    IHeader1 = IHeader1 & " - " & Chr(10) & RYear
    AHeader1 = AHeader1 & " - " & Chr(10) & RYear
    FHeader1 = FHeader1 & " - " & Chr(10) & RYear
```

```
    BHeader1 = BHeader1 & " - " & Chr(10) & RYear
    CHeader1 = CHeader1 & " - " & Chr(10) & RYear
    DHeader1 = DHeader1 & " - " & Chr(10) & RYear
    KHeader1 = KHeader1 & " - " & Chr(10) & RYear
Else
    IHeader1 = IHeader1 & " - " & RYear
    AHeader1 = AHeader1 & " - " & RYear
    FHeader1 = FHeader1 & " - " & RYear
    BHeader1 = BHeader1 & " - " & RYear
    CHeader1 = CHeader1 & " - " & RYear
    DHeader1 = DHeader1 & " - " & RYear
    KHeader1 = KHeader1 & " - " & RYear
End If

'Parse K Report components
iComma = InStr(KComps, ",")
NumComps = Val(Left(KComps, iComma - 1))
If NumComps > 0 Then
    KRptOK = True
    ReDim KRptComps(NumComps)
    KRptComps(0) = NumComps
    For I = 1 To NumComps
        KComps = Right(KComps, Len(KComps) - iComma)
        iComma = InStr(KComps, ",")
        If iComma > 0 Then
            KRptComps(I) = Val(Left(KComps, iComma - 1))
        Else
            KRptComps(I) = Val(KComps)
        End If
    Next
Else
    KRptOK = False
    ErrDat.Book = "N/A"
    ErrDat.Sheet = "N/A"
    ErrDat.Row = "N/A"
    ErrDat.Level = 3
    ErrDat.Desc = "No components for K Report."
    Call RecordError
End If

'Parse components to display decimals in I Report
iComma = InStr(IPrintDecimals, ",")
```

```
NumComps = Val(Left(IPrintDecimals, iComma - 1))
If NumComps > 0 Then
    IRptOK = True
    ReDim IDecimals(NumComps)
    IDecimals(0) = NumComps
    For I = 1 To NumComps
        IPrintDecimals = Right(IPrintDecimals, Len(IPrintDecimals) - iComma)
        iComma = InStr(IPrintDecimals, ",")
        If iComma > 0 Then
            IDecimals(I) = Val(Left(IPrintDecimals, iComma - 1))
        Else
            IDecimals(I) = Val(IPrintDecimals)
        End If
    Next
Else
    IRptOK = False
    ErrDat.Book = "N/A"
    ErrDat.Sheet = "N/A"
    ErrDat.Row = "N/A"
    ErrDat.Level = 3
    ErrDat.Desc = "No components for I Report."
    Call RecordError
End If
End Sub
```

```
'Run the Reports

Sub Reports()
Dim RptFileName As String
Dim OpenResult As Integer
Dim I As Integer, J As Integer, iRow As Integer
Dim IHeader3 As String
Dim iPeriod As String
Dim SheetName As String
Dim ThisCS As Single
Dim StrCS As String
Dim ThisComp As Integer
Dim ThisClass As Integer
Dim LastCol As Integer
Dim LastRow As Integer
Dim iBracket As String
Dim ThisTitle As String
Dim iDec As Integer
Dim RevCS As Single

'Open a new file for the report
If iModule = 1 Then
    RptFileName = ModName & ".IRpt.xls"
    RevCS = ComponentData(CompIndx(IRevComp)).SubSegment
Else
    RptFileName = ModName & "." & ModuleLabel(iModule) & "Rpt.xls"
End If
Call OpenWB(RptFileName, ModPath, OpenResult)
If OpenResult <> 2 Then
    'Workbook already exists, so clear out the worksheets
    For I = 1 To Sheets.Count
        Sheets(I).Activate
        Cells.Select
        Selection.Clear
        Cells.PageBreak = x1PageBreakNone
    Next
End If

'Put the components on the report, adding a worksheet if necessary
'This section is now generalized so it can be used for all reports,
'except A, F, and B reports will need documentation row as well.
For I = 1 To MaxCompon
```

```
If ComponentData(I).Label <> "" Then
  If ComponentData(I).CS <= LastCS Then
      ThisCS = ComponentData(I).CS
      StrCS = Right("00" & ThisCS, 2)
  Else
      ThisCS = ComponentData(I).SubSegment
      StrCS = Trim(Str(ThisCS))
  End If

  ThisComp = Val(Trim(ComponentData(I).Label))

  'Get the right field for components to print
  If iModule = 1 Then
      PrintMod = ComponentData(I).PrintI
      Application.StatusBar = "Running Input Report"
  ElseIf iModule = 2 Then
      PrintMod = ComponentData(I).PrintA
      Application.StatusBar = "Running A Report"
  ElseIf iModule = 3 Then
      PrintMod = ComponentData(I).PrintF
      Application.StatusBar = "Running Factor Report"
  ElseIf iModule = 4 Then
      PrintMod = ComponentData(I).PrintB
      Application.StatusBar = "Running B (PESSA) Report"
      KRpt = False
  ElseIf iModule = 5 Then
      PrintMod = ComponentData(I).PrintC
      Application.StatusBar = "Running C (Cost Segments and Components) Report"
      DRpt = False
  End If
  'Create a worksheet if needed
  If ((Trim(UCase(PrintMod)) = "BOTH") Or _
      (Trim(UCase(PrintMod)) = "CS") Or _
      (USPSorPRC = 1 And Trim(UCase(PrintMod)) = "USPS") Or _
      (USPSorPRC = 2 And Trim(UCase(PrintMod)) = "PRC")) And _
      ComponentData(I).CS <> CalcKeysCS Then
      'Create a page for this Cost Segment if not already there

      SheetName = "CS" & StrCS

      On Error GoTo NoSheet
      EFlag = 0
```

```
Sheets(SheetName).Activate
On Error GoTo NoSheet
If EFlag > 0 Then
    Sheets.Add
    ActiveSheet.Name = SheetName
    J = Sheets.Count
    Sheets(SheetName).Move After:=Sheets(J)
    'Put mail classes on the page if components are displayed in rows
    If ThisCS >= IStartPrintRows And iModule = 1 Then
        Cells(RptStartRow, RptStartCol).Select
        Selection.Value = "Factor Name"
        Cells(RptStartRow, RptStartCol + 1).Select
        Selection.Value = "Factor Number"
        Cells(RptStartRow, RptStartCol + 2).Select
        Selection.Value = "Factor Value"
    'Otherwise components are displayed in the usual column format
    Else
        Call RptClasses
    End If
End If

'Put the component data on the page if going in rows
If ThisCS >= IStartPrintRows And iModule = 1 Then
    'Report is in row format
    'Find the last row on the page for component data
    Cells(xlMaxLine, RptStartCol + 1).Select
    Selection.End(xlUp).Select
    LastRow = Selection.Row
    If LastRow < RptStartRow + 1 Then
        LastRow = RptStartRow + 1
    End If
    Cells(LastRow + 1, RptStartCol).Select
    ThisTitle = ComponentData(I).Title
    iBracket = InStr(ThisTitle, "[")
    If iBracket > 0 Then
        ThisTitle = Left(ThisTitle, iBracket - 1)
    End If
    Selection.Value = ThisTitle
    Cells(LastRow + 1, RptStartCol + 1).Select
    Selection.Value = "'(" & ThisComp & ")"
    Selection.HorizontalAlignment = x1Center
    Cells(LastRow + 1, RptStartCol + 2).Select
```

```
    Selection.Value = CostMatrix(TotClass, ThisComp)
    Selection.NumberFormat = "#,##0"
    Columns("A:C").EntireColumn.AutoFit

Else  'Going in column format
    'Find the last used column
    Cells(RptCompRow, 255).Select
    Selection.End(xlToLeft).Select
    LastCol = Selection.Column
    If LastCol < RptStartCol + 1 Then
        LastCol = RptStartCol + 1
    End If

    'Find the last row on the page with mail class data
    Cells(xlMaxLine, RptStartCol).Select
    Selection.End(xlUp).Select
    LastRow = Selection.Row

    Cells(RptStartRow, LastCol + 1).Select
    Selection.ColumnWidth = 14
    Selection.WrapText = True
    'Look to see if [] in component name, if so, then eliminate that part of title
    ThisTitle = ComponentData(I).Title
    iBracket = InStr(ThisTitle, "[")
    If iBracket > 0 Then
        ThisTitle = Left(ThisTitle, iBracket - 1)
    End If
    Selection.Value = ThisTitle
    Selection.HorizontalAlignment = xlCenter
    Cells(RptStartRow + 1, LastCol + 1).Select

    Selection.Value = "'(" & ThisComp & ")"
    Selection.HorizontalAlignment = xlCenter
    Cells(RptStartRow + 2, LastCol + 1).Select
    Selection.Value = ComponentData(I).SubSegment
    Selection.HorizontalAlignment = xlCenter

    'Put in documentation for appropriate module
    If iModule > 1 And iModule < 5 Then
        Cells(RptStartRow + 3, LastCol + 1).Select
        Call RptDoc(I) 'Notice appropriate cell already selected on cost segment sheet
    End If
```

```
            'Put cost matrix data onto the worksheet
            For iRow = RptStartRow + 4 To LastRow
                ThisClass = Val(Trim(Cells(iRow, RptStartCol + 1).Value))
                Cells(iRow, LastCol + 1).Select
                If ThisClass > 0 Then
                    Selection.Value = CostMatrix(ThisClass, ThisComp)
                    Selection.NumberFormat = "#,##0"
                    If iModule = 1 Then
                        For iDec = 1 To IDecimals(0)
                            If IDecimals(iDec) = ThisComp Then
                                Selection.NumberFormat = "0.00000000"
                            End If
                        Next
                    End If
                End If
            Next
            'Now calculate the variability by putting a formula in the cell
            'Don't do this for now

            'ThisFormula = "=if(R" & LastRow - 1 & "C" & LastCol + 1 & " <> 0, R" & LastRow - 3 & "C" & LastCol + 1 & "
/ R" & LastRow - 1 & "C" & LastCol + 1 & ", 0)"
            'Cells(LastRow, LastCol + 1).Select
            'Selection.Formula = ThisFormula
            'Selection.Style = "Percent"
            'Selection.NumberFormat = "0.0%"
        End If
    End If
    'Create data to the CS Summary worksheet worksheet if needed
    If Trim(UCase(PrintMod)) = "CS" Then
        'Create a page for the Cost Segment summary
        SheetName = "CSSummary"

        On Error GoTo NoSheet
        EFlag = 0
        Sheets(SheetName).Activate
        On Error GoTo NoSheet
        If EFlag > 0 Then
            Sheets.Add
            ActiveSheet.Name = SheetName
            Sheets(SheetName).Move Before:=Sheets(1)
            'Put mail classes on the page if components are displayed in rows
```

```
        Call RptClasses
    End If

    'Find the last used column
    Cells(RptCompRow, 255).Select
    Selection.End(xlToLeft).Select
    LastCol = Selection.Column
    If LastCol < RptStartCol + 1 Then
        LastCol = RptStartCol + 1
    End If

    'Find the last row on the page with mail class data
    Cells(xlMaxLine, RptStartCol).Select
    Selection.End(xlUp).Select
    LastRow = Selection.Row

    Cells(RptStartRow, LastCol + 1).Select
    Selection.ColumnWidth = 14
    Selection.WrapText = True
    'Look to see if [] in component name, if so, then eliminate that part of title
    ThisTitle = ComponentData(I).Title
    iBracket = InStr(ThisTitle, "[")
    If iBracket > 0 Then
        ThisTitle = Left(ThisTitle, iBracket - 1)
    End If
    Selection.Value = ThisTitle
    Selection.HorizontalAlignment = xlCenter
    Cells(RptStartRow + 1, LastCol + 1).Select

    Selection.Value = "'(" & ThisComp & ")"
    Selection.HorizontalAlignment = xlCenter

    'Put cost matrix data onto the worksheet
    For iRow = RptStartRow + 4 To LastRow
        ThisClass = Val(Trim(Cells(iRow, RptStartCol + 1).Value))
        Cells(iRow, LastCol + 1).Select
        If ThisClass > 0 Then
            Selection.Value = CostMatrix(ThisClass, ThisComp)
            Selection.NumberFormat = "#,##0"
        End If
    Next
End If
```

```
  End If
Next

'If this is the I Report and IRevComp > 0 Then add a worksheet to display the Revenue component
If iModule = 1 And IRevComp > 0 And iRunType <> 3 Then
    SheetName = "Revenue"
    On Error GoTo NoSheet
    EFlag = 0
    Sheets(SheetName).Activate
    On Error GoTo NoSheet
    If EFlag > 0 Then
        Sheets.Add
        ActiveSheet.Name = SheetName
        J = Sheets.Count
        StrCS = "CS" & Trim(Str(RevCS))
        Sheets(SheetName).Move After:=Sheets(StrCS)
        Call RptRevClasses
    End If
    'Find the last used column
    Cells(RptCompRow, 255).Select
    Selection.End(xlToLeft).Select
    LastCol = Selection.Column
    If LastCol < RptStartCol + 1 Then
        LastCol = RptStartCol + 1
    End If

    'Find the last row on the page with mail class data
    Cells(xlMaxLine, RptStartCol).Select
    Selection.End(xlUp).Select
    LastRow = Selection.Row

    Cells(RptStartRow, LastCol + 1).Select
    Selection.ColumnWidth = 14
    Selection.WrapText = True
    ThisTitle = ComponentData(CompIndx(IRevComp)).Title
    Selection.Value = ThisTitle
    Selection.HorizontalAlignment = xlCenter
    Cells(RptStartRow + 1, LastCol + 1).Select

    Selection.Value = "'(" & IRevComp & ")"
    Selection.HorizontalAlignment = xlCenter
    Cells(RptStartRow + 2, LastCol + 1).Select
```

```
    Selection.Value = ComponentData(I).SubSegment
    Selection.HorizontalAlignment = xlCenter

    'Put cost matrix data for revenue component onto the worksheet
    For iRow = RptStartRow + 4 To LastRow
        ThisClass = Val(Trim(Cells(iRow, RptStartCol + 1).Value))
        Cells(iRow, LastCol + 1).Select
        If ThisClass > 0 Then
            Selection.Value = CostMatrix(ThisClass, IRevComp)
            Selection.NumberFormat = "#,##0"
        End If
    Next
End If

'Delete Sheet1, Sheet2, Sheet3 from workbook
Application.DisplayAlerts = False
'This will generate Error 9 (subscript out of range) if the specified worksheet does not exist
On Error GoTo NoSheet
EFlag = 0
Sheets("Sheet1").Select
On Error GoTo 0
If EFlag = 0 Then
    Worksheets("Sheet1").Delete
End If
On Error GoTo NoSheet
EFlag = 0
Sheets("Sheet2").Select
On Error GoTo 0
If EFlag = 0 Then
    Worksheets("Sheet2").Delete
End If
On Error GoTo NoSheet
EFlag = 0
Sheets("Sheet3").Select
On Error GoTo 0
If EFlag = 0 Then
    Worksheets("Sheet3").Delete
End If
Application.DisplayAlerts = True

'Format the report for printing (ala Roll Forward).  Need separate formatting
'for printing in rows and printing in columns.
```

```
'Get the headers to footers set, row or column format, and page breaks put in.
'Add headers and footers to report

Application.ScreenUpdating = True
For J = 1 To Sheets.Count
    StrCS = Right(Sheets(J).Name, Len(Sheets(J).Name) - 2)
    If Val(StrCS) <= LastCS And Sheets(J).Name <> "Revenue" Then
        Header3 = "Cost Segment " & StrCS
        If Val(StrCS) > 0 Then
            Header3 = Header3 & " / " & CostSegment(CSIndex(Val(StrCS))).CSName
        End If
        Worksheets(Sheets(J).Name).Activate
        If iModule = 1 Then
            Call ColFmtBorders(IHeader1, IHeader2, Header3)
        ElseIf iModule = 2 Then
            Call ColFmtBorders(AHeader1, AHeader2, Header3)
        ElseIf iModule = 3 Then
            Call ColFmtBorders(FHeader1, FHeader2, Header3)
        ElseIf iModule = 4 Then
            Call ColFmtBorders(BHeader1, BHeader2, Header3)
        ElseIf iModule = 5 Then
            Call ColFmtBorders(CHeader1, CHeader2, Header3)
        End If
    Else
    'Get the distribution key cost segment name for use in header
        For I = 1 To CompsMax
            If Trim(StrCS) = ComponentData(I).SubSegment Then
                Header3 = ComponentData(I - 1).Title
                Exit For
            End If
        Next
        If Val(StrCS) >= IStartPrintRows And iModule = 1 Then
            Worksheets(Sheets(J).Name).Activate
            Call RowFmtBorders(IHeader1, IHeader2, Header3)
        Else
            Worksheets(Sheets(J).Name).Activate
            RevSheet = False
            If iModule = 1 And Sheets(J).Name <> "Revenue" Then
                Call ColFmtBorders(IHeader1, IHeader2, Header3)
            ElseIf iModule = 1 And Sheets(J).Name = "Revenue" Then
                RevSheet = True
                Header3 = "RPW Inputs"
```

```
                Call ColFmtBorders(IHeader1, IHeader2, Header3)
                RevSheet = False
            ElseIf iModule = 2 Then
                Call ColFmtBorders(AHeader1, AHeader2, Header3)
            ElseIf iModule = 3 Then
                Call ColFmtBorders(FHeader1, FHeader2, Header3)
            ElseIf iModule = 4 Then
                Call ColFmtBorders(BHeader1, BHeader2, Header3)
            ElseIf iModule = 5 Then
                Call ColFmtBorders(CHeader1, CHeader2, Header3)
            End If
        End If
    End If
Next
Application.ScreenUpdating = True
Windows(CRABook).Activate
Sheets("CRA Menu").Activate
Application.GoTo ("ModelName")
Application.ScreenUpdating = False


Exit Sub

NoSheet:
EFlag = 1
Resume Next

End Sub
```

```
Public Sub RptClasses()
'Inserts the Mail Classes onto the current sheet in the location specified
'by public variables RptStartRow and RptStartCol

Dim I, R As Long

For I = 1 To ClassMax
  R = RptStartRow + I + 3
  Cells(R, RptStartCol).Select
  Prefix = ""
  If MailClassData(I).Display1 = 2 Or MailClassData(I).Display1 = 4 Then
    Prefix = "    "
  ElseIf MailClassData(I).Display1 = 3 Then
    Prefix = "        "
  End If
  Selection.Value = Prefix & MailClassData(I).Name
  If MailClassData(I).Display1 = 1 Or MailClassData(I).Display1 = 4 Then
    Selection.Font.Bold = True
  End If

  If MailClassData(I).Number > 0 Then
    Cells(R, RptStartCol + 1).Select
    Selection.Value = MailClassData(I).Number
  End If
Next

'Don't do this for now.
'Cells(R + 1, RptStartCol).Select
'Selection.Value = "Variability"
'Selection.Font.Bold = True

Cells(RptStartRow, RptStartCol).Select
Selection.Value = "Component Name"
Selection.Font.Bold = True
Selection.Font.Italic = True
Cells(RptStartRow + 1, RptStartCol).Select
Selection.Value = "Component Number"
Selection.Font.Bold = True
Selection.Font.Italic = True
Cells(RptStartRow + 2, RptStartCol).Select
Selection.Value = "Cost Segment"
Selection.Font.Bold = True
```

```
Selection.Font.Italic = True

Columns("A:B").Select
Columns("A:B").EntireColumn.AutoFit

If iModule > 1 And iModule < 5 Then
    Cells(RptStartRow + 3, RptStartCol).Select
    Selection.Value = "Notes"
    With Selection
        .HorizontalAlignment = xlLeft
        .VerticalAlignment = xlTop
        .WrapText = True
    End With
    Selection.Font.Bold = True
    Selection.Font.Italic = True
End If

Cells(1, 1).Select

End Sub
```

```
Public Sub RptRevClasses()
'Inserts the Revenue Classes onto the current sheet in the location specified
'by public variables RptStartRow and RptStartCol

Dim I, R As Long

For I = 1 To RevMax
  R = RptStartRow + I + 3
  Cells(R, RptStartCol).Select
  Prefix = ""
  If RevClassData(I).Display1 = 2 Or RevClassData(I).Display1 = 4 Then
    Prefix = "    "
  ElseIf RevClassData(I).Display1 = 3 Then
    Prefix = "        "
  End If
  Selection.Value = Prefix & RevClassData(I).Name
  If RevClassData(I).Display1 = 1 Or RevClassData(I).Display1 = 4 Then
    Selection.Font.Bold = True
  End If

  If RevClassData(I).Number > 0 Then
    Cells(R, RptStartCol + 1).Select
    Selection.Value = RevClassData(I).Number
  End If
Next

Cells(RptStartRow, RptStartCol).Select
Selection.Value = "Component Name"
Selection.Font.Bold = True
Selection.Font.Italic = True
Cells(RptStartRow + 1, RptStartCol).Select
Selection.Value = "Component Number"
Selection.Font.Bold = True
Selection.Font.Italic = True

Columns("A:B").Select
Columns("A:B").EntireColumn.AutoFit

Cells(1, 1).Select

End Sub
```

```
Sub ColFmtBorders(Header1, Header2, Header3)
'Puts in borders for worksheets where component data is printed in columns.
'Assumes worksheet is active
'Header1 and Header2 are user-defined in model workbook
'Header3 is the cost segment number and name

Dim iClass As Integer, Border As Integer, Class As Integer
Dim LastRow As Integer, LastCol As Integer
Dim ColCnt As Integer
Dim ColtoCheck As Integer
Dim PageStr As String
Dim MaxNum As Integer

Application.ScreenUpdating = False
'Draw a border around the whole thing
Cells(RptStartRow, RptStartCol).Select
'Find the last column and the last row on the worksheet
Cells(RptCompRow, 255).Select
Selection.End(xlToLeft).Select
LastCol = Selection.Column
If LastCol < RptStartCol + 1 Then
    LastCol = RptStartCol + 1
End If

'Find the last row on the page with mail class data
Cells(xlMaxLine, RptStartCol).Select
Selection.End(xlUp).Select
LastRow = Selection.Row

Range(Cells(RptStartRow, RptStartCol), Cells(LastRow, LastCol)).Select
Selection.Borders(xlDiagonalDown).LineStyle = xlNone
Selection.Borders(xlDiagonalUp).LineStyle = xlNone
With Selection.Borders(xlEdgeLeft)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
```

```
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeRight)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlInsideVertical)
    .LineStyle = xlContinuous
    .Weight = xlHairline
    .ColorIndex = xlAutomatic
End With
Selection.Borders(xlInsideHorizontal).LineStyle = xlNone

'Draw a border line before mail class data starts
Cells(RptStartRow + 3, RptStartCol).Select
Range(Selection, Cells(RptStartRow + 3, LastCol)).Select
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With

'Mail class data goes on worksheet as iClass + RptStartRow + 3
If RevSheet = True Then
    MaxNum = RevMax
Else
    MaxNum = ClassMax
End If

For iClass = 1 To MaxNum
    If RevSheet = True Then
        Border = RevClassData(iClass).Border
        Class = RevClassData(iClass).Number
    Else
        Border = MailClassData(iClass).Border
        Class = MailClassData(iClass).Number
    End If
    'Find row for this mail class
```

```
    ThisRow = iClass + RptStartRow + 3

    If Border = 1 Then
        Cells(ThisRow, RptStartCol).Select
        Range(Selection, Cells(ThisRow, LastCol)).Select
        With Selection.Borders(xlEdgeBottom)
            .LineStyle = xlContinuous
            .Weight = xlHairline
            .ColorIndex = xlAutomatic
        End With
    End If
    If Border = 2 Then
        Cells(ThisRow, RptStartCol).Select
        Range(Selection, Cells(ThisRow, LastCol)).Select
        With Selection.Borders(xlEdgeBottom)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
    End If
Next

'Do page set up, using Header1 - Header3 as sent to subroutine
If iModule = 1 Then
    PageStr = "Page I-&P"
ElseIf iModule = 2 Then
    PageStr = "Page A-&P"
ElseIf iModule = 3 Then
    PageStr = "Page F-&P"
ElseIf iModule = 4 And KRpt = False Then
    PageStr = "Page B-&P"
ElseIf iModule = 4 And KRpt = True Then
    PageStr = "Page K-&P"
ElseIf iModule = 5 And DRpt = False Then
    PageStr = "Page C-&P"
ElseIf iModule = 5 And DRpt = True Then
    PageStr = "Page D-&P"
End If
Application.ScreenUpdating = True
With ActiveSheet.PageSetup
    .LeftHeader = "&""Arial,Bold""&11" & Chr(10) & Chr(10) & Chr(10) & Header3
    .CenterHeader = "&""Arial,Bold""&12" & Header1 & Chr(10) & Header2
```

```
    .CenterFooter = PageStr
    .RightFooter = "&8Printed On: &D &T"
    .TopMargin = Application.InchesToPoints(1.2)
    .HeaderMargin = Application.InchesToPoints(0.75)
    .BottomMargin = Application.InchesToPoints(0.5)
    .FooterMargin = Application.InchesToPoints(0.35)
    .LeftMargin = Application.InchesToPoints(0.5)
    .RightMargin = Application.InchesToPoints(0.5)
    .PrintTitleColumns = "$A:$B"
    .PrintTitleRows = "$1:$5"
    .Orientation = xlLandscape
    .FirstPageNumber = xlAutomatic
    .Zoom = 74
    .FitToPagesWide = False
    .FitToPagesTall = False
End With
Application.ScreenUpdating = False

'Add page breaks after every 10 columns of data after title columns.
ColCnt = 10  'Number of columns on each page, not including print titles in first 2 columns.
ColtoCheck = ColCnt + 2
Cells(RptCompRow, ColtoCheck + 1).Select
ThisColComp = Selection.Value
Do Until ThisColComp = 0
    ActiveWindow.SelectedSheets.VPageBreaks.Add Before:=ActiveCell
    ColtoCheck = ColtoCheck + ColCnt
    Cells(RptCompRow, ColtoCheck + 1).Select
    ThisColComp = Selection.Value
Loop

If RevSheet = True Then
    'Put in page break after 49 rows on page
    Rows(50).Select
    ActiveWindow.SelectedSheets.HPageBreaks.Add Before:=ActiveCell
End If

Rows(RptStartRow).Select
Selection.VerticalAlignment = xlTop

End Sub
```

```
Sub RowFmtBorders(Header1, Header2, Header3)
'Use this page formatting routine for factor "cost" segments that will be printed
'in rows, with factor name and factor number as the columns
'Assumes worksheet is active
'Header1 and Header2 are user-defined in model workbook
'Header3 is the factor "cost segment" name

Dim LastRow As Integer

Application.ScreenUpdating = False
'Find the last column and the last row on the worksheet
LastCol = RptStartCol + 2

Cells(xlMaxLine, RptStartCol).Select
Selection.End(xlUp).Select
LastRow = Selection.Row

'Draw a border around the whole thing
Cells(RptStartRow, RptStartCol).Select
Range(Selection, Cells(LastRow, LastCol)).Select
Selection.Borders(xlDiagonalDown).LineStyle = xlNone
Selection.Borders(xlDiagonalUp).LineStyle = xlNone
With Selection.Borders(xlEdgeLeft)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeRight)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
```

```
With Selection.Borders(xlInsideVertical)
    .LineStyle = xlContinuous
    .Weight = xlHairline
    .ColorIndex = xlAutomatic
End With
Selection.Borders(xlInsideHorizontal).LineStyle = xlNone

'Draw a border line before factor data starts
Cells(RptStartRow + 1, RptStartCol).Select
Range(Selection, Cells(RptStartRow + 1, LastCol)).Select
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With

'If worksheet has more than 45 rows in use then put in a page break and a border
Cells(xlMaxLine, RptStartCol).Select
Selection.End(xlUp).Select
LastRow = Selection.Row
If LastRow > 45 Then
    Cells(45, RptStartCol).Select
    Range(Selection, Cells(45, LastCol)).Select
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    Cells(46, RptStartCol).Select
    ActiveWindow.SelectedSheets.HPageBreaks.Add Before:=ActiveCell
End If

Application.ScreenUpdating = True
'Do page set up, using Header1 - Header3 as sent to subroutine
With ActiveSheet.PageSetup
    .LeftHeader = "&""Arial,Bold""&11" & Chr(10) & Chr(10) & Chr(10) & Header3
    .CenterHeader = "&""Arial,Bold""&12" & Header1 & Chr(10) & Header2
    .CenterFooter = "Page I-&P"
    .RightFooter = "&8Printed On: &D &T"
    .TopMargin = Application.InchesToPoints(1.2)
    .HeaderMargin = Application.InchesToPoints(0.75)
    .BottomMargin = Application.InchesToPoints(0.6)
```

```
    .FooterMargin = Application.InchesToPoints(0.35)
    .LeftMargin = Application.InchesToPoints(0.5)
    .RightMargin = Application.InchesToPoints(0.5)
    .PrintTitleRows = "$1:$3"
    .Orientation = xlLandscape
    .FirstPageNumber = xlAutomatic
    .Zoom = 74
    .FitToPagesWide = False
    .FitToPagesTall = False
End With


End Sub
```

```
Sub RptDoc(Idx As Integer)
'This subroutine puts the documentation in the selected cell
Dim Routine As String
Dim Distrib As Integer
Dim Misc As Integer
If iModule = 2 Then
    Routine = ComponentData(Idx).ModuleA
    Distrib = ComponentData(Idx).DistribA
    Misc = ComponentData(Idx).MiscA
ElseIf iModule = 3 Then
    Routine = ComponentData(Idx).ModuleF
    Distrib = ComponentData(Idx).DistribF
    Misc = ComponentData(Idx).MiscF
ElseIf iModule = 4 Then
    Routine = ComponentData(Idx).ModuleB
    Distrib = ComponentData(Idx).DistribB
    Misc = ComponentData(Idx).MiscB
End If

If Routine = "AV" Then
    StrDoc = "Compute variability = " & ComponentData(Idx).ManualVariability
ElseIf Routine = "DA" Or Routine = "DV" Or Routine = "DX" Or Routine = "DZ" Then
    StrDoc = "Distrib. on " & Distrib
ElseIf Routine = "CV" Then
    StrDoc = "Compute cl. 199 on " & Distrib
ElseIf Routine = "CP" Then
    StrDoc = "Comp. " & Distrib & " - comp. " & Misc
ElseIf Routine = "ST" Then
    StrDoc = "Set total to " & Distrib
ElseIf Routine = "SA" Then
    StrDoc = "Set equal to " & Distrib
ElseIf Routine = "AJ" Then
    StrDoc = "Adjust using factor " & Distrib
End If

Selection.Value = StrDoc
With Selection
    .HorizontalAlignment = xlCenter
    .VerticalAlignment = xlTop
    .WrapText = True
End With
End Sub
```

```
Sub KReport()
'Special report not done with the generalized subroutine

Dim RptFileName As String
Dim OpenResult As Integer
Dim I As Integer, iRow As Integer
Dim SheetName As String
Dim ThisComp As Integer
Dim ThisClass As Integer
Dim LastCol As Integer
Dim LastRow As Integer
Dim ThisTitle As String

'Open a new file for the report
RptFileName = ModName & ".KRpt.xls"

Call OpenWB(RptFileName, ModPath, OpenResult)
If OpenResult <> 2 Then
    'Workbook already exists, so clear out the worksheets
    For I = 1 To Sheets.Count
        Sheets(I).Activate
        Cells.Select
        Selection.Clear
        Cells.PageBreak = x1PageBreakNone
    Next
End If

SheetName = "KReport"
On Error GoTo NoSheet
EFlag = 0
Sheets(SheetName).Activate
On Error GoTo NoSheet
If EFlag > 0 Then
    Sheets.Add
    ActiveSheet.Name = SheetName
    J = Sheets.Count
    Sheets(SheetName).Move After:=Sheets(J)
    'Put mail classes on the page if components are displayed in ro
    Call RptClasses
End If
```

```
'Put the K Report components on the worksheet

Application.StatusBar = "Running K Report"
KRpt = True

For I = 1 To KRptComps(0)

    ThisComp = Val(KRptComps(I))
    'Find the last used column
    Cells(RptCompRow, 255).Select
    Selection.End(xlToLeft).Select
    LastCol = Selection.Column
    If LastCol < RptStartCol + 1 Then
        LastCol = RptStartCol + 1
    End If

    'Find the last row on the page with mail class data
    Cells(xlMaxLine, RptStartCol).Select
    Selection.End(xlUp).Select
    LastRow = Selection.Row

    Cells(RptStartRow, LastCol + 1).Select
    Selection.ColumnWidth = 14
    Selection.WrapText = True
    'Look to see if [] in component name, if so, then eliminate that part of title
    ThisTitle = ComponentData(CompIndx(ThisComp)).Title
    iBracket = InStr(ThisTitle, "[")
    If iBracket > 0 Then
        ThisTitle = Left(ThisTitle, iBracket - 1)
    End If
    Selection.Value = ThisTitle
    Selection.HorizontalAlignment = xlCenter
    Cells(RptStartRow + 1, LastCol + 1).Select

    Selection.Value = "'(" & ThisComp & ")"
    Selection.HorizontalAlignment = xlCenter
'
    'Put cost matrix data onto the worksheet

    For iRow = RptStartRow + 4 To LastRow
        ThisClass = Val(Trim(Cells(iRow, RptStartCol + 1).Value))
        Cells(iRow, LastCol + 1).Select
```

```
        If ThisClass > 0 Then
            Selection.Value = CostMatrix(ThisClass, ThisComp)
            Selection.NumberFormat = "#,##0"
        End If
    Next
Next

'Delete Sheet1, Sheet2, Sheet3 from workbook
Application.DisplayAlerts = False
'This will generate Error 9 (subscript out of range) if the specified worksheet does not exist
On Error GoTo NoSheet
EFlag = 0
Sheets("Sheet1").Select
On Error GoTo 0
If EFlag = 0 Then
    Worksheets("Sheet1").Delete
End If
On Error GoTo NoSheet
EFlag = 0
Sheets("Sheet2").Select
On Error GoTo 0
If EFlag = 0 Then
    Worksheets("Sheet2").Delete
End If
On Error GoTo NoSheet
EFlag = 0
Sheets("Sheet3").Select
On Error GoTo 0
If EFlag = 0 Then
    Worksheets("Sheet3").Delete
End If
Application.DisplayAlerts = True

'Format the report for printing (ala Roll Forward).  Need separate formatting
'for printing in rows and printing in columns.
'Get the headers to footers set, row or column format, and page breaks put in.
'Add headers and footers to report

Application.ScreenUpdating = True
Sheets(SheetName).Activate

Call ColFmtBorders(KHeader1, KHeader2, KHeader3)
```

```
Application.ScreenUpdating = True
Windows(CRABook).Activate
wsMenu.Activate
Application.ScreenUpdating = False

Exit Sub

NoSheet:
EFlag = 1
Resume Next

End Sub
```

```
Sub DReport()
'Generates the D Report
'Special report not done with the generalized subroutine

Dim RptFileName As String
Dim OpenResult As Integer
Dim I As Integer, iRow As Integer
Dim SheetName As String
Dim ThisComp As Integer
Dim ThisClass As Integer
Dim LastCol As Integer
Dim LastRow As Integer
Dim ThisTitle As String

'Open a new file for the report
RptFileName = ModName & ".DRpt.xls"

Call OpenWB(RptFileName, ModPath, OpenResult)
If OpenResult <> 2 Then
    'Workbook already exists, so clear out the worksheets
    For I = 1 To Sheets.Count
        Sheets(I).Activate
        Cells.Select
        Selection.Clear
        Cells.PageBreak = x1PageBreakNone
    Next
End If

SheetName = "DReport"
On Error GoTo NoSheet
EFlag = 0
Sheets(SheetName).Activate
On Error GoTo NoSheet
If EFlag > 0 Then
    Sheets.Add
    ActiveSheet.Name = SheetName
    J = Sheets.Count
    Sheets(SheetName).Move After:=Sheets(J)
    'Put mail classes on the page if components are displayed in ro
    Call RptClasses
End If
```

```
'Put the D Report components on the worksheet

Application.StatusBar = "Running D Report"
DRpt = True

For I = 1 To DMax
  'Put this component on the report page if appropriate
  If (Trim(UCase(DData(I).DPrint)) = "BOTH") Or _
     (USPSorPRC = 1 And Trim(UCase(DData(I).DPrint)) = "USPS") Or _
     (USPSorPRC = 2 And Trim(UCase(DData(I).DPrint)) = "PRC") Or _
     (iRunType = 3 And Trim(UCase(DData(I).DRFOnly) = "BOTH") Or _
     (iRunType = 3 And USPSorPRC = 1 And Trim(UCase(DData(I).DRFOnly)) = "USPS") Or _
     (iRunType = 3 And USPSorPRC = 2 And Trim(UCase(DData(I).DRFOnly)) = "PRC")) Then

        ThisComp = Val(DData(I).DComp)
        'Find the last used column
        Cells(RptCompRow, 255).Select
        Selection.End(xlToLeft).Select
        LastCol = Selection.Column
        If LastCol < RptStartCol + 1 Then
            LastCol = RptStartCol + 1
        End If

        'Find the last row on the page with mail class data
        LastRow = ClassRow(TotClass)

        Cells(RptStartRow, LastCol + 1).Select
        Selection.ColumnWidth = 14
        Selection.WrapText = True
        'Look to see if [] in component name, if so, then eliminate that part of title
        ThisTitle = ComponentData(CompIndx(ThisComp)).Title
        iBracket = InStr(ThisTitle, "[")
        If iBracket > 0 Then
            ThisTitle = Left(ThisTitle, iBracket - 1)
        End If
        Selection.Value = ThisTitle
        Selection.HorizontalAlignment = xlCenter
        Cells(RptStartRow + 1, LastCol + 1).Select

        Selection.Value = "'(" & ThisComp & ")"
        Selection.HorizontalAlignment = xlCenter
'
```

```
        'Put cost matrix data onto the worksheet

        For iRow = RptStartRow + 4 To LastRow
            ThisClass = Val(Trim(Cells(iRow, RptStartCol + 1).Value))
            Cells(iRow, LastCol + 1).Select
            If ThisClass > 0 Then
                Selection.Value = CostMatrix(ThisClass, ThisComp)
                Selection.NumberFormat = "#,##0"
            End If
        Next
    End If
Next

'Delete Sheet1, Sheet2, Sheet3 from workbook
Application.DisplayAlerts = False
'This will generate Error 9 (subscript out of range) if the specified worksheet does not exist
On Error GoTo NoSheet
EFlag = 0
Sheets("Sheet1").Select
On Error GoTo 0
If EFlag = 0 Then
    Worksheets("Sheet1").Delete
End If
On Error GoTo NoSheet
EFlag = 0
Sheets("Sheet2").Select
On Error GoTo 0
If EFlag = 0 Then
    Worksheets("Sheet2").Delete
End If
On Error GoTo NoSheet
EFlag = 0
Sheets("Sheet3").Select
On Error GoTo 0
If EFlag = 0 Then
    Worksheets("Sheet3").Delete
End If
Application.DisplayAlerts = True

'Format the report for printing (ala Roll Forward).  Need separate formatting
'for printing in rows and printing in columns.
'Get the headers to footers set, row or column format, and page breaks put in.
```

```
'Add headers and footers to report

Application.ScreenUpdating = True
Sheets(SheetName).Activate

DHeader3 = ""
Call ColFmtBorders(DHeader1, DHeader2, DHeader3)

Application.ScreenUpdating = True
Windows(CRABook).Activate
wsMenu.Activate
Application.ScreenUpdating = False

Exit Sub

NoSheet:
EFlag = 1
Resume Next

End Sub
```

F. Writing Output Workbooks – ModWorkbooks

```
'Contains subroutines that deal with setting up workbooks

Public Sub WriteClasses()
'Inserts the Mail Classes onto the current sheet in the location specified
'by public variables SeqCol, ClassNameCol, ClassNoCol, and FirstRow

Dim I, R As Long
Dim MC As Variant
Dim Msg, Prefix As String
Dim Seq As Integer

R = FirstRow - 1
Seq = 0
Cells(ClassLabelRow, SeqCol).Select
Selection.Value = "Row" & vbLf & "Seq"
Cells(ClassLabelRow, ClassNameCol).Select
Selection.Value = "Mail Class or Service"
Cells(ClassLabelRow, ClassNoCol).Select
Selection.Value = "Class" & vbLf & "No."
For I = 1 To ClassMax
  R = R + 1
  Cells(R, ClassNameCol).Select
  Prefix = ""
  If MailClassData(I).Display1 = 2 Then
    Prefix = "    "
  ElseIf MailClassData(I).Display1 = 3 Or MailClassData(I).Display1 = 5 Then
    Prefix = "        "
  ElseIf MailClassData(I).Display1 = 4 Then
    Prefix = "            "
  End If
  Selection.Value = Prefix & MailClassData(I).Name
  If (MailClassData(I).Display1 = 1 Or MailClassData(I).Display1 = 5) Then
    Selection.Font.Bold = True
  End If
'Write RowSeq number only if Class > 0
  If MailClassData(I).Number > 0 Then
    Seq = Seq + 1
    Cells(R, SeqCol).Select
    Selection.Value = Seq
```

```
    Cells(R, ClassNoCol).Select
    Selection.Value = MailClassData(I).Number
  End If
Next

Columns("A:C").Select
Columns("A:C").EntireColumn.AutoFit

Cells(1, 1).Select

End Sub
```

```
Sub MakeOutputMatrixSheet()

'Assumes we are already in the right workbook.
'Simply copies InputMatrix to OutputMatrix and initializes the MtxChange array

'Test to see if we already have an "OutputMatrix" sheet
'This will generate Error 9 (subscript out of range) if the specified worksheet does not exist
OutMatrixSheetName = "OutputMatrix"
EFlag = 0
On Error GoTo OutputMatrixError
Sheets(OutMatrixSheetName).Activate
On Error GoTo 0
'If it is already there, delete it
If EFlag = 0 Then
  Application.DisplayAlerts = False
  Sheets(OutMatrixSheetName).Delete
  Application.DisplayAlerts = True
End If
'Now copy the InputMatrix to the end of workbook and rename it OutputMatrix
Sheets("InputMatrix").Select
Sheets("InputMatrix").Copy After:=Sheets(Sheets.Count)
ActiveSheet.Name = "OutputMatrix"
Call InitMtxChange

Exit Sub

OutputMatrixError:
EFlag = 1
Resume Next

End Sub
```

```
Public Sub WriteComps(CS)
'Puts component names and numbers onto the current worksheet
'in the order they appear on the Master Component list.

Dim LastCol As Integer
Dim CompDoc As String
Dim iClass As Integer
Dim CFormula As String

On Error GoTo 0

'Initialize to the last used column
LastCol = FirstColumn - 1

'Turns off warnings about numbers formatted as text
Application.ErrorCheckingOptions.NumberAsText = False

For I = 1 To CompsMax
    'Use non-subtotal components only
    'or subtotal components for cost-containing cost segments
    'i.e., don't add subtotals on distribution key cost segments
  If ComponentData(I).CS = CS Then
    LastCol = LastCol + 1
    Columns(LastCol).ColumnWidth = 15
    Cells(CompNameRow, LastCol).Select
    Selection.Value = ComponentData(I).Title
    Selection.WrapText = True
    Selection.HorizontalAlignment = xlCenter
    Cells(CompNoRow, LastCol).Select
    Selection.Value = "'" & Right("0000" & Trim(Str(ComponentData(I).Label)), 4)
    Selection.HorizontalAlignment = xlCenter
    Cells(DocRow, LastCol).Select
    Selection.HorizontalAlignment = xlCenter
    CompColumn(Val(ComponentData(I).Label)) = LastCol
    CompSheet(Val(ComponentData(I).Label)) = ActiveSheet.Name
    If (ComponentData(I).SubTotalA And iModule = 2) Or _
       (ComponentData(I).SubTotalF And iModule = 3) Or _
       (ComponentData(I).SubTotalB And iModule = 4) Or _
       (ComponentData(I).SubTotalC And iModule = 4) Then
      Selection.Value = "Subtotal"
    Else
      Distrib = 0
```

```
      Misc = 0
      CompDoc = ""
      If iModule = 2 Then
        CompDoc = ComponentData(I).ModuleA
        Distrib = ComponentData(I).DistribA
        Misc = ComponentData(I).MiscA
      ElseIf iModule = 3 Then
        CompDoc = ComponentData(I).ModuleF
        Distrib = ComponentData(I).DistribF
        Misc = ComponentData(I).MiscF
      ElseIf iModule = 4 Then
        CompDoc = ComponentData(I).ModuleB
        Distrib = ComponentData(I).DistribB
        Misc = ComponentData(I).MiscB
      End If
      If Distrib > 0 Then
        CompDoc = CompDoc & " " & Right("0000" & Trim(Str(Distrib)), 4)
        If Misc > 0 Then
          CompDoc = CompDoc & " (" & Right("0000" & Trim(Str(Misc)), 4) & ")"
        End If
      End If
      If CompDoc = "AV" Then
        CompDoc = CompDoc & " " & ComponentData(I).ManualVariability
      End If
      Selection.Value = CompDoc
    End If
'Put formulas into each cell to get data from InputMatrix
    For iClass = 1 To ClassMax
      If MailClassData(iClass).Number > 0 Then
        CFormula = "=InputMatrix!R" & Trim(Str(Val(ComponentData(I).Label) + MatrixRowOffset)) & "C" &
Trim(Str(MailClassData(iClass).Number + MatrixColOffset))
        Cells(iClass + FirstRow - 1, LastCol).Select
        Selection.Formula = CFormula
        Selection.NumberFormat = "#,##0"
      End If
    Next
  End If
Next
Rows(CompNameRow).EntireRow.AutoFit

End Sub
```

```
Public Sub UpdateOutputMatrix()
'Updates sheet OutputMatrix for calculations performed in this workbook
'For each element of MtxChange that is set to TRUE, it re-writes that portion of the corresponding component
'for each valid Mail Class (column) in the array.

Dim Cmp, Cls As Integer
Dim ThisClass As Integer

Application.StatusBar = ModuleLabel(iModule) & " Workbook:  Updating Output Matrix"
For Cmp = 1 To MaxComps
  If MtxChange(Cmp) Then
    For Cls = 1 To ClassMax
      ThisClass = MailClassData(Cls).Number
      If ThisClass > 0 Then
        Sheets(CompSheet(Cmp)).Select
        Cells(ClassRow(ThisClass), CompColumn(Cmp)).Select
        CostMatrix(ClassIndx(Cls), Cmp) = Selection.Value
      End If
    Next
  End If
Next

Sheets("OutputMatrix").Select
For Cmp = 1 To MaxComps
  If MtxChange(Cmp) Then
    For Cls = 1 To ClassMax
      ThisClass = MailClassData(Cls).Number
      If ThisClass > 0 Then
        Cells(Cmp + MatrixRowOffset, ThisClass + MatrixColOffset).Select
        Selection.Value = CostMatrix(ClassIndx(Cls), Cmp)
        Selection.NumberFormat = "#,##0"
      End If
    Next
  End If
  MtxChange(Cmp) = False
Next

End Sub
```

```
Sub OpenWB(OpenBook, OpenPath, Result As Integer)

'Input argument OpenBook is the name of the workbook to be opened.
'Input argument OpenPath is the path where the workbook to be opened is located.
'Output argument Result is set as follows:
'  0 = workbook was already open before calling
'  1 = workbook already existed, and is now open
'  2 = workbook had to be created
'In all cases, the workbook will be open and will be the active workbook upon exiting this sub

'FilI holds the next available file number
'EFlag is an error flag
Dim FilI, EFlag As Integer

'FullBookName is the concatenated path and workbook name
Dim FullBookName As String

'OldStatusText is the text that was displayed in the statusbar before coming here
Dim OldStatusText As String

EFlag = 0
Result = 0
If Len(OpenPath) > 0 Then
  FullBookName = OpenPath & OpenBook
Else
  FullBookName = CRAPath & OpenBook
End If

'Check to see whether the file needs to be created
'Try to open the workbook.  This generates an error (which is trapped) if it doesn't exist
'This uses a DOS-style Open statement rather than an Excel (Workbooks.Open) to test whether
'the file exists.  FreeFile is a function that returns the next available file number.
OldStatusText = Application.StatusBar
Application.StatusBar = strCurYear & "Checking for file " & FullBookName
FilI = FreeFile
On Error GoTo OpenWBErr
Open FullBookName For Input As FilI
Close FilI
On Error GoTo 0
If EFlag > 0 Then
  Workbooks.Add
  ActiveWorkbook.SaveAs Filename:=FullBookName, _
```

```
                        FileFormat:=xlNormal, _
                        Password:="", _
                        WriteResPassword:="", _
                        ReadOnlyRecommended:=False, _
                        CreateBackup:=False
  Result = 2
Else

'If we get here, the file exists.  Check to see if it is already open
'This will generate Error 9 (subscript out of range) if the workbook is not open
'Note that this routine will not distinguish between the desired workbook and
'one with the same name from a different folder that is already open.
'If there is another workbook with the same name already open, this routine will
'think that the desired workbook is open already.
  On Error GoTo OpenWBErr
  Windows(OpenBook).Activate
  On Error GoTo 0
  If EFlag > 0 Then
    Application.StatusBar = strCurYear & "Opening " & FullBookName
'The second paramter of the Open method (False) tells Excel NOT TO UPDATE LINKS
'Of course, there shouldn't be any links in these workbooks.
    Workbooks.Open FullBookName, False
    Result = 1
  End If
End If

OpenSave:
'All exits from the Sub go through here to track whether to add workbook to
'list of workbooks that need to be saved and closed at the end
If Result > 0 Then
  SCCount = SCCount + 1
  ReDim Preserve SaveCloseNames(SCCount)
  ReDim Preserve SaveCloseFlag(SCCount)
  SaveCloseNames(SCCount) = OpenBook
  SaveCloseFlag(SCCount) = SCFlag
  SCFlag = True
End If

Application.StatusBar = OldStatusText
Exit Sub

OpenWBErr:
```

```
EFlag = 1
Resume Next

End Sub
```

```
Sub AddCSSheets()
'Adds Cost Segment worksheets and places Mail Class labels on each sheet
'Make sheet name for each Cost Segment
'Also makes one sheet for each of the "additional" sheets listed on worksheet CostSegments

Dim LastRow As Long
Dim SheetName As String
Dim EFlag As Integer
Dim CS, CSNum As Integer
Dim FirstCSSheet As String
Dim StatTxt As String
Dim I As Integer

StatTxt = ModuleLabel(iModule) & " Workbook: Creating worksheet for "
FirstCSSheet = ""
For CS = 1 To CSMax
    CSNum = CostSegment(CS).CSNumber
    StrCS = Right("00" & Trim(CSNum), 2)
    SheetName = "CS" & StrCS
    If FirstCSSheet = "" Then FirstCSSheet = SheetName
    Application.StatusBar = StatTxt & SheetName
    Workbooks(WorkingBook).Activate
'This will generate Error 9 (subscript out of range) if the specified worksheet does not exist
    On Error GoTo NoSheet
    EFlag = 0
    Sheets(SheetName).Activate
    On Error GoTo 0
    If EFlag > 0 Then
      Sheets.Add
      ActiveSheet.Name = SheetName
      J = Sheets.Count
      Sheets(SheetName).Move After:=Sheets(J)
    Else
'This makes sure there's nothing on the page before we start
      Cells.Select
      Selection.ClearContents
    End If


'Put the mail classes onto this sheet
    CSSheet = SheetName
    Call WriteClasses
```

```
    Columns("A:C").Select
    Columns("A:C").EntireColumn.AutoFit

    Cells(1, 1).Select
    Selection.Value = "Cost Segment " & CostSegment(CS).CSNumber & ": " & CostSegment(CS).CSName
    Selection.Font.Color = vbBlue
    Selection.Font.Bold = True
    Selection.Font.Size = 12
    Rows("1:1").RowHeight = 20

'Place common headers on worksheet
    Call CSSheetCommonHeaders
'Add components to worksheet
    Call WriteComps(CSNum)
    Cells(1, 1).Select
Next

'Add the "Cost Segement Place-holder" and other sheets
For I = 1 To PlaceHolderCount
  SheetName = PlaceHolderSheets(I)
  Application.StatusBar = StatTxt & SheetName
  On Error GoTo NoSheet
  EFlag = 0
  Sheets(SheetName).Activate
  On Error GoTo 0
  If EFlag > 0 Then
    Sheets.Add
    ActiveSheet.Name = SheetName
    J = Sheets.Count
    Sheets(SheetName).Move After:=Sheets(J)
  Else
'This makes sure there's nothing on the page before we start
    Cells.Select
    Selection.ClearContents
  End If
  CSSheet = SheetName
  Call WriteClasses
  Columns("A:C").Select
  Columns("A:C").EntireColumn.AutoFit
  Cells(1, 1).Select
  Selection.Value = PlaceHolderTypes(I)
  Selection.Font.Color = vbBlue
```

```
  Selection.Font.Bold = True
  Selection.Font.Size = 12
  Rows("1:1").RowHeight = 20
  Call CSSheetCommonHeaders
  Call FillDKSheet
  Cells(1, 1).Select
Next


'Go back to the first CS sheet
Sheets(FirstCSSheet).Activate

Exit Sub

NoSheet:
EFlag = 1
Resume Next

End Sub
```

```
Public Sub CSSheetCommonHeaders()
'Places the common headers on the current worksheet
Rows(CompNameRow).RowHeight = 25
Rows(CompNameRow).WrapText = True
Cells(CompNameRow, ColHeadCol).Select
Selection.Value = "Component Name"
Selection.HorizontalAlignment = xlRight
Selection.WrapText = False
Cells(CompNoRow, ColHeadCol).Select
Selection.Value = "Component Number"
Selection.HorizontalAlignment = xlRight
Cells(DocRow, ColHeadCol).Select
Selection.Value = "Documentation"
Selection.HorizontalAlignment = xlRight

End Sub
```

```
Public Sub MakeWorkingBook()
'Creates or opens the WorkingBook
'Uses various Public variables to build the book name

Dim Result As Integer

WorkingBook = ModName & "." & ModuleLabel(iModule) & ".xls"
Application.StatusBar = "Creating " & WorkingBook & " in " & ModPath
Call OpenWB(WorkingBook, ModPath, Result)
Call MakeInputMatrixSheet
Call AddCSSheets
Call MakeOutputMatrixSheet
Call ClassSubtotals

End Sub
```

```
Public Sub MakeInputMatrixSheet()
'Creates a sheet named InputMatrix in the current workbook
'Deletes any other sheets in the workbook (if any)
'Writes the current Cost Matrix onto InputMatrix

Dim SheetName As String
Dim EFlag As Integer
Dim I, ShCount As Integer
Dim R As Long
Dim C As Integer

Application.StatusBar = "Creating InputMatrix sheet in " & WorkingBook

'Part 1: Create or Clear the contents of sheet InputMatrix
SheetName = "InputMatrix"
Workbooks(WorkingBook).Activate
'This will generate Error 9 (subscript out of range) if the specified worksheet does not exist
On Error GoTo NoIMatrix
EFlag = 0
Sheets(SheetName).Activate
On Error GoTo 0
If EFlag > 0 Then
  Sheets.Add
  ActiveSheet.Name = SheetName
  J = Sheets.Count
  Sheets(SheetName).Move After:=Sheets(J)
Else
'This makes sure there's nothing on the page before we start
  Cells.Select
  Selection.ClearContents
End If

'Part 2: Delete all the other sheets in this workbook
ShCount = Sheets.Count
For I = ShCount To 1 Step -1
  If Sheets(I).Name <> "InputMatrix" Then
    Application.DisplayAlerts = False
    Sheets(I).Delete
    Application.DisplayAlerts = True
  End If
Next
```

```
'Part 3: Write the Cost Matrix (and labels) on sheet InputMatrix
'The writing of labels is not active yet, just the writing of data
'This puts Class Number in Col 1, Class Name in Col 2
For C = 1 To MaxClass
  For R = 1 To MaxComps
    If CostMatrix(C, R) <> 0 Then
      Cells(R + MatrixRowOffset, C + MatrixColOffset).Select
      Selection.Value = CostMatrix(C, R)
    End If
  Next
Next

Exit Sub

NoIMatrix:
EFlag = 1
Resume Next

End Sub
```

```
Public Sub FillDKSheet()
'This sub is specific to the Place-holder (DK) worksheets
'Puts component names and numbers onto the current worksheet
'in the order they appear on the Master Component list.

Dim LastCol As Integer
Dim CompDoc As String
Dim SSComp As String
Dim LComp As Integer

On Error GoTo 0

'Initialize to the last used column
LastCol = FirstColumn - 1

'Turns off warnings about numbers formatted as text
Application.ErrorCheckingOptions.NumberAsText = False

SSComp = Trim(Mid(ActiveSheet.Name, 3))
LComp = Len(SSComp)

For I = 1 To CompsMax
    'Use non-subtotal components only
    'or subtotal components for cost-containing cost segments
    'i.e., don't add subtotals on distribution key cost segments
  If Left(ComponentData(I).SubSegment, LComp) = SSComp Then
    LastCol = LastCol + 1
    Columns(LastCol).ColumnWidth = 15
    Cells(CompNameRow, LastCol).Select
    Selection.Value = ComponentData(I).Title
    Selection.WrapText = True
    Selection.HorizontalAlignment = xlCenter
    Cells(CompNoRow, LastCol).Select
    Selection.Value = "'" & Right("0000" & Trim(Str(ComponentData(I).Label)), 4)
    Selection.HorizontalAlignment = xlCenter
    Cells(DocRow, LastCol).Select
    Selection.HorizontalAlignment = xlCenter
    CompColumn(Val(ComponentData(I).Label)) = LastCol
    CompSheet(Val(ComponentData(I).Label)) = ActiveSheet.Name

    If (ComponentData(I).SubTotalA And iModule = 2) Or _
       (ComponentData(I).SubTotalF And iModule = 3) Or _
```

```
       (ComponentData(I).SubTotalB And iModule = 4) Or _
       (ComponentData(I).SubTotalC And iModule = 4) Then
      Selection.Value = "Subtotal"
    Else
      Distrib = 0
      Misc = 0
      CompDoc = ""
      If iModule = 2 Then
        CompDoc = ComponentData(I).ModuleA
        Distrib = ComponentData(I).DistribA
        Misc = ComponentData(I).MiscA
      ElseIf iModule = 3 Then
        CompDoc = ComponentData(I).ModuleF
        Distrib = ComponentData(I).DistribF
        Misc = ComponentData(I).MiscF
      ElseIf iModule = 4 Then
        CompDoc = ComponentData(I).ModuleB
        Distrib = ComponentData(I).DistribB
        Misc = ComponentData(I).MiscB
      End If
      If Distrib > 0 And Distrib <> "AV" Then
        CompDoc = CompDoc & " " & Trim(Str(Distrib))
        If Misc > 0 Then
          CompDoc = CompDoc & " (" & Trim(Str(Misc)) & ")"
        End If
      End If
      If CompDoc = "AV" Then
        CompDoc = CompDoc & " " & ComponentData(I).ManualVariability
      End If
      Selection.Value = CompDoc
      'Put formulas into each cell to get data from InputMatrix.  N.Kay added 1/13/05
      For iClass = 1 To ClassMax
        If MailClassData(iClass).Number > 0 Then
          CFormula = "=InputMatrix!R" & Trim(Str(Val(ComponentData(I).Label) + MatrixRowOffset)) & "C" &
Trim(Str(MailClassData(iClass).Number + MatrixColOffset))
          Cells(iClass + FirstRow - 1, LastCol).Select
          Selection.Formula = CFormula
          Selection.NumberFormat = "#,##0"
        End If
      Next
    End If
  End If
```

```
Next
Rows(CompNameRow).EntireRow.AutoFit

End Sub
```

G. Response to User Actions – ModUserInterface

```
Dim StartTime, EndTime As Date

Sub SelectModel()
Dim ModelIndex As Integer

Application.GoTo ("ModelName")
ModelIndex = Selection.Value
Application.GoTo ("ModelsSheet")
Cells(ModelIndex + 1, 1).Select
ModName = Selection.Value
Cells(ModelIndex + 1, 2).Select
ModPath = Trim(Selection.Value)
If Right(ModPath, 1) <> "\" Then ModPath = ModPath & "\"
Application.GoTo ("ModelName")

End Sub
```

```
Sub ChangeRunType()
'If the user selects "Roll Forward", then make sure cost matrix is specified
Dim Msg As Variant

Application.GoTo ("RunType")
iRunType = Selection.Value
If iRunType = 3 Then
    'Get the control file
    RFCntlFileBrowse
End If

End Sub
```

```
Sub Run()
Dim Msg As String
Dim TabLabel As String

StartTime = Time
ErrRow = 2
Application.GoTo ("ErrorsSheet")
ErrSheet = ActiveSheet.Name
Call InitializeErrorSheet

'Checks for presence of all named ranges
Call CheckPhase1
If ErrMax > 0 Then
  Msg = "Initialization error(s) encountered." & vbCrLf & vbCrLf _
      & "One or more named ranges not found.  See Errors sheet." & vbCrLf & vbCrLf _
      & "Cannot continue."
  MsgBox Msg, vbOKOnly + vbCritical, "Initialization error"
  Exit Sub
End If

'Read the names of all the worksheets in the CRA workbook into variables
On Error GoTo GoErr
MenuSheet = ActiveSheet.Name
TabLabel = "Input Table Sheet"
Application.GoTo ("InputTableSheet")
ControlSheet = ActiveSheet.Name
TabLabel = "Lists Sheet"
Application.GoTo ("ListsSheet")
ListSheet = ActiveSheet.Name
TabLabel = "Limits Sheet"
Application.GoTo ("LimitsSheet")
LimitSheet = ActiveSheet.Name

On Error GoTo GoErr2

ErrDat.Desc = "SaveClose label cannot be found."
ErrDat.Sheet = MenuSheet
Application.GoTo ("SaveClose")
ynSaveClose = Selection.Value

On Error GoTo 0
```

```
Call Driver
Call RunCleanup

Exit Sub

GoErr:
ErrDat.Book = ActiveWorkbook.Name
ErrDat.Desc = TabLabel & " cannot be identified by its named range."
ErrDat.Level = 3
If ErrDat.Sheet = "" Then
  ErrDat.Sheet = "(Unidentified Worksheet)"
End If
Call RecordError
ErrDat.Sheet = ""
Resume Next

GoErr2:
ErrDat.Book = ActiveWorkbook.Name
ErrDat.Level = 3
Call RecordError
ErrDat.Sheet = ""
ErrDat.Desc = ""
Resume Next

End Sub
```

```
Sub RunCleanup()
Dim DeltaT As Single
Dim RMsg As String

Call SaveCloseWorkbooks
If ErrStop Then
  Windows(CRABook).Activate
  Application.GoTo ("ModelName")
  Application.StatusBar = False
  Application.ScreenUpdating = True
  EndTime = Time
'This expresses run-time in seconds
  DeltaT = (EndTime - StartTime) * 86400
  RMsg = "Run aborted in " & CLng(DeltaT) & " seconds because of errors encountered during error-check."
  RMsg = RMsg & vbCrLf & vbCrLf & ErrRow - 3 & " errors encountered." & vbCrLf
  RMsg = RMsg & "Highest error level encountered: " & ErrMax & vbCrLf
  RMsg = RMsg & "You specified abort if error is higher than " & ErrMaxAllow
  MsgBox RMsg, vbOKOnly, "Run aborted"
ElseIf iRunType = 1 Then
  Windows(CRABook).Activate
  Application.GoTo ("ModelName")
  Application.StatusBar = False
  Application.ScreenUpdating = True
  EndTime = Time
'This expresses run-time in seconds
  DeltaT = (EndTime - StartTime) * 86400
  RMsg = "Data check completed in " & CLng(DeltaT) & " seconds."
  MsgBox RMsg, vbOKOnly, "Done"
Else
  Windows(CRABook).Activate
  Application.GoTo ("ModelName")
  Application.StatusBar = False
  Application.ScreenUpdating = True
  EndTime = Time
'This expresses run-time in seconds
  DeltaT = (EndTime - StartTime) * 86400
  RMsg = "Run completed in " & CLng(DeltaT) & " seconds."
  MsgBox RMsg, vbOKOnly, "Done"
End If

End Sub
```

```
Sub GetRFOptions()
'Get Rollforward options from user

Dim C As Integer
Dim ThisResult As Integer
Dim InputFil As String
Dim InputDir As String
Dim FileToOpen As String
Dim GetFile As Boolean

Application.ScreenUpdating = True
GetFile = True
RunOK = True

Do
    FileToOpen = Application.GetOpenFilename("Excel files (*.xls), *.xls", , "Select TY File")
    If FileToOpen = "False" Then
        MsgBox "No TY file selected", vbOKOnly + vbInformation, "Aborting Run"
        RunOK = False
        Exit Sub
    Else
        GetFile = False
    End If
    RevF = StrReverse(FileToOpen)
    iSlash = InStr(RevF, "\")
    If iSlash > 0 Then
        InputFil = Right(FileToOpen, iSlash - 1)
        InputDir = Left(FileToOpen, Len(FileToOpen) - iSlash + 1)
    End If
    If GetFile = False Then
        MsgResult = MsgBox("You have selected   " & InputFil & ".  Is this correct?", vbYesNo, "Confirm TY File
Selection")
        If MsgResult = vbNo Then
            GetFile = True
        Else
            GetFile = False
        End If
    End If
Loop Until GetFile = False

MsgResult = MsgBox("Have you entered the Space, Rental, Capital, and Alaska Air factors into " & InputFil & "?", _
            vbYesNo, "Confirm Factors Entered")
```

```
If MsgResult = vbNo Then
    MsgBox "Enter factors before running reports", vbOKOnly, "Aborting Run"
    RunOK = False
    Exit Sub
End If

'Loads a model and makes it "active"

Application.StatusBar = ""
Application.ScreenUpdating = False

'Open the cost matrix file and load into the CostMatrix array
Call OpenWB(InputFil, InputDir, ThisResult)
If ThisResult = 2 Then
    MsgBox "Model " & InputFil & " could not be found in directory " & Chr(10) _
        & InputDir, vbCritical, "Critical Error!"
    Exit Sub
End If

'Set global variable for model path to TY file directory
ModPath = InputDir

'Read the OutputMatrix worksheet in opened workbook and copy into cost matrix array
Sheets("OutputMatrix").Activate
For C = SkipClass To MaxClass
  Application.StatusBar = "Reading Input Matrix, class " & C & " of " & MaxClass
  For R = 1 To MaxComps
    Cells(R + MatrixRowOffset, C + MatrixColOffset).Select
    CostMatrix(C, R) = Selection.Value
  Next
Next

'Get RF Options needed for Reports

FormResult = 0
RFOptions.TextBox1.Text = ""
RFOptions.TextBox2.Text = ""
RFOptions.TextBox3.Text = "0.0"
RFOptions.TextBox1.SetFocus
Application.ScreenUpdating = True
wsMenu.Activate
RFOptions.Show
```

```
If FormResult = 1 Then
    RFOptions.Hide
    wsMenu.Activate
    Application.ScreenUpdating = False
    MsgBox "You must enter Rollforward Options", vbOKOnly, "Aborting Run"
    RunOK = False
    Exit Sub
End If

wsMenu.Activate
Application.ScreenUpdating = False

End Sub
```

```
Sub CancelForm()
'Called when terminate button pressed on a user form.
FormResult = 1

End Sub

Sub CancelRFOptions()
'Called when Cancel button or terminate button pressed on a RF Options form.
FormResult = 1
RFOptions.Hide

End Sub

Sub EndRFOptions()
'FINISHED button pressed on RfOptions form.

RateCase = UCase(RFOptions.TextBox1.Text)
RFYear = RFOptions.TextBox2.Text
Contingency = RFOptions.TextBox3.Text
If RFOptions.OptionButton1 = True Then
    Rfwd = 1
ElseIf RFOptions.OptionButton2 = True Then
    Rfwd = 2
End If

If RFOptions.OptionButton3 = True Then
    VerType = 1
ElseIf RFOptions.OptionButton4 = True Then
    VerType = 2
End If

RFOptions.Hide
Application.ScreenUpdating = True

End Sub
```

```
Sub RFCntlFileBrowse()
Dim FileToOpen As String
Dim CntlFil As String
Dim CntlDir As String
Dim GetFile As Boolean
Dim MsgResult As Integer
Dim iSlash As Integer
Dim iPeriod As Integer

RunOK = True
GetFile = True
Do
    FileToOpen = Application.GetOpenFilename("Excel files (*.xls), *.xls", , "Select Control File")
    If FileToOpen = "False" Then
        MsgBox "No control file selected", vbOKOnly + vbInformation, "Aborting Run"
        RunOK = False
        Exit Sub
    End If

    RevF = StrReverse(FileToOpen)
    iSlash = InStr(RevF, "\")
    If iSlash > 0 Then
        CntlFil = Right(FileToOpen, iSlash - 1)
        CntlDir = Left(FileToOpen, Len(FileToOpen) - iSlash + 1)
    End If
    MsgResult = MsgBox("You have selected   " & CntlFil & ".  Is this correct?", vbYesNo, "Confirm Control File
Selection")
    If MsgResult = vbNo Then
        GetFile = True
    Else
        GetFile = False
    End If

Loop Until GetFile = False

iPeriod = InStr(CntlFil, ".")
If iPeriod > 0 Then
    ModName = Left(CntlFil, iPeriod - 1)
Else
    ModName = CntlFil
End If
```

```
ModPath = CntlDir
CntlBook = CntlFil

End Sub
```

H. Utility Functions – ModUtil

```
Public Function IsClass(C) As Boolean
'The argument, C, is a variant.
'If C is numeric and is the value of a valid mail class, the function returns True.
'Otherwise it returns False

Dim I As Integer

IsClass = False

If Not IsNumeric(C) Then
  Exit Function
End If

For I = 1 To ClassMax
  If C = MailClassData(I).Number Then
    IsClass = True
    Exit For
  End If
Next

End Function
```

```
Public Function IsComponent(C) As Boolean
'The argument, C, is a variant.
'If C is numeric and is the value of a valid component, the function returns True.
'Otherwise it returns False

Dim I As Integer

IsComponent = False

If Not IsNumeric(C) Then
  Exit Function
End If

For I = 1 To CompsMax
  If C = Val(ComponentData(I).Label) Then
    IsComponent = True
    Exit For
  End If
Next

End Function
```

```
Sub FindText(Txt As String)
'Finds the text specified.  Selected range must already be set
Selection.Find(what:=Txt, _
               After:=ActiveCell, _
               LookIn:=xlFormulas, _
               LookAt:=xlPart, _
               SearchOrder:=xlByRows, _
               SearchDirection:=xlNext, _
               MatchCase:=False, _
               SearchFormat:=False).Select
End Sub
```

```
Sub SaveCloseWorkbooks()
'Saves and closes workbooks that the program had to open or create.
'Workbooks that were not opened by the program are not closed.
Dim iWB As Integer

'ynSaveClose is the global flag that indicates whether user wants workbooks saved and closed
If Not ynSaveClose Then
  Exit Sub
End If


For iWB = 1 To SCCount
  On Error GoTo CloseErr
  Application.StatusBar = "Checking " & Workbooks(SaveCloseNames(iWB)).Name
  Windows(SaveCloseNames(iWB)).Activate
  On Error GoTo 0
  If (ActiveWorkbook.Saved) Or (Not SaveCloseFlag(iWB)) Then
    Application.StatusBar = "Closing " & Workbooks(SaveCloseNames(iWB)).Name
  Else
    Application.StatusBar = "Saving and closing " & Workbooks(SaveCloseNames(iWB)).Name
    ActiveWorkbook.Save
  End If
  ActiveWorkbook.Close
NextBook:
  On Error GoTo 0
Next

Exit Sub

CloseErr:
Resume NextBook

End Sub
```

I. Error Handling Routines – ModErrorHandling

```
Sub RecordError()
'Displays an error on the Errors page in the main workbook
Dim WB, WS As String
'Save the name of the workbook and sheet we came from
WB = ActiveWorkbook.Name
WS = ActiveSheet.Name
'Write the error on the Errors worksheet
Workbooks(CRABook).Activate
Application.GoTo ("ErrorsSheet")
Cells(ErrRow, 1).Select
Selection.Value = ErrRow - 2
Cells(ErrRow, 2).Select
Selection.Value = ErrDat.Level
Cells(ErrRow, 3).Select
Selection.Value = ErrDat.Book
Cells(ErrRow, 4).Select
Selection.Value = ErrDat.Sheet
Cells(ErrRow, 5).Select
Selection.Value = ErrDat.Row
Cells(ErrRow, 6).Select
Selection.Value = ErrDat.Desc

'Track highest error level encountered
If ErrDat.Level > ErrMax Then
  ErrMax = ErrDat.Level
End If

'Increment row pointer
ErrRow = ErrRow + 1

'Clear data
ErrDat.Book = ""
ErrDat.Desc = ""
ErrDat.Level = 0
ErrDat.Row = 0
ErrDat.Sheet = ""

'Return to where we came from
Workbooks(WB).Activate
```

```
Sheets(WS).Activate
End Sub
```

```
Sub InitializeErrorSheet()
Dim WB, WS As String
Dim R2 As Long
Dim R As String

WB = ActiveWorkbook.Name
WS = ActiveSheet.Name
'Go to the errors sheet and clear all previous info
Application.GoTo ("ErrorsSheet")
Cells(xlMaxLine, 1).Select
Selection.End(xlUp).Select
R2 = Selection.Row
R = "A2:F" & R2
Range(R).Select
Selection.Clear
Cells(2, 1).Select
Selection.Value = 0
Cells(2, 2).Select
Selection.Value = 0
Cells(2, 3).Select
Selection.Value = "Run started"
Cells(2, 4).Select
Selection.Value = Date
Cells(2, 6).Select
Selection.Value = Time
ErrRow = 3

ErrMax = 0
ErrStop = False

'Return to where we came from
Workbooks(WB).Activate
Sheets(WS).Activate

End Sub
```

```
Sub DataCheck()
'Driver routine for performing data logic / consistency checks

Dim DKFlag As Boolean
Dim dT As Double
Dim iClass As Integer
Dim iComp As Integer
Dim iDetail As Integer
Dim iDK As Integer
Dim iiComp As Integer
Dim Msg As String
Dim T0 As Double
Dim T1 As Double

T0 = Timer

Workbooks(CRABook).Activate
Application.StatusBar = "Performing Data Check"

'Component loop
For iComp = 1 To CompsMax
'****************
'See if there is a duplicate
  For iiComp = iComp + 1 To CompsMax
    If Val(ComponentData(iiComp).Label) > 0 Then
      If Val(ComponentData(iiComp).Label) = Val(ComponentData(iComp).Label) Then
        ErrDat.Level = 2
        ErrDat.Book = CntlBook
        ErrDat.Sheet = CntlCompSheet
        ErrDat.Row = ComponentData(iComp).Row
        ErrDat.Desc = "Duplicate component listing for component " & ComponentData(iComp).Label
        Call RecordError
        Exit For
      End If
    End If
  Next
'****************
'Parent/child relationship for each routine must be logical (in the same C/S)
  If ComponentData(iComp).ParentA > 0 Then
    If ComponentData(iComp).CS <> ComponentData(CompIndx(Val(ComponentData(iComp).Label))).CS Then
      ErrDat.Level = 2
      ErrDat.Book = CntlBook
```

```
      ErrDat.Sheet = CntlCompSheet
      ErrDat.Row = ComponentData(iComp).Row
      ErrDat.Desc = "'A' routine Parent of Component " & ComponentData(iComp).Label & " (" & ComponentData(iComp).CS _
                  & ") is in a different Cost Segment"
      Call RecordError
    End If
  End If
'****************
'Every argument must be a valid Component or Class
'Module1 (Pre-A) DK
  If ComponentData(iComp).Distrib1 > 0 And Not IsComponent(ComponentData(iComp).Distrib1) Then
    Msg = "Module1 DK or source"
    Call NonCompArg(iComp, Msg)
  End If
'Module1 (Pre-A) additional parameter
  If ComponentData(iComp).Misc1 > 0 And Not IsClass(ComponentData(iComp).Misc1) Then
    Msg = "Module1 additional parameter"
    Call NonCompArg(iComp, Msg)
  End If
'Module A DK
  If ComponentData(iComp).DistribA > 0 And Not IsComponent(ComponentData(iComp).DistribA) Then
    Msg = "Module A DK or source"
    Call NonCompArg(iComp, Msg)
  End If
'Module A additional parameter
  If ComponentData(iComp).MiscA > 0 And Not IsComponent(ComponentData(iComp).MiscA) Then
    Msg = "Module A additional parameter"
    Call NonCompArg(iComp, Msg)
  End If
'Module F DK
  If ComponentData(iComp).DistribF > 0 And Not IsComponent(ComponentData(iComp).DistribF) Then
    Msg = "Module F DK or source"
    Call NonCompArg(iComp, Msg)
  End If
'Module F additional parameter
  If ComponentData(iComp).MiscF > 0 And Not IsComponent(ComponentData(iComp).MiscF) Then
    Msg = "Module F additional parameter"
    Call NonCompArg(iComp, Msg)
  End If
'Module B DK
  If ComponentData(iComp).DistribB > 0 And Not IsComponent(ComponentData(iComp).DistribB) Then
    Msg = "Module B DK or source"
```

```
    Call NonCompArg(iComp, Msg)
  End If
'Module B additional parameter
  If ComponentData(iComp).MiscB > 0 And Not IsComponent(ComponentData(iComp).MiscB) Then
    Msg = "Module B additional parameter"
    Call NonCompArg(iComp, Msg)
  End If
'Arguments for each routine must meet the following criteria
'AV Variability > 0
'DA Arg1 is valid DK, Arg2 is empty
'DV Arg1 is valid DK, Arg2 is empty
'DZ Arg1 is valid DK, Arg2 is valic component in 98.3
'DX Arg1 is valid DK, Arg2 is valid component in 21 or 98.3
'CV Arg1 is valid DK, Arg2 is empty
'CP Arg1 is valid component in same subsegment, Arg2 is valid component in same subsegment, but <> Arg1
'ST Arg1 is valid component in same subsegment, Arg2 is empty
'SA Arg1 is valid non-subtotal component, Arg2 is empty
'AJ Arg1 is valid component in 98.3, Arg2 is valid Class
'Module1 (GDEI) parameters
  If Len(Trim(ComponentData(iComp).Module1)) > 0 And ComponentData(iComp).Module1 <> "AJ" Then
    ErrDat.Book = CntlBook
    ErrDat.Sheet = CntlCompSheet
    ErrDat.Row = ComponentData(iComp).Row
    ErrDat.Level = 2
    ErrDat.Desc = "Module 1: the only routine that is allowed is AJ. " & ComponentData(iComp).Module1 & " is not
allowed."
    Call RecordError
  ElseIf Len(Trim(ComponentData(iComp).Module1)) > 0 Then
    Call CheckRoutine(iComp, ComponentData(iComp).Module1, ComponentData(iComp).Distrib1, ComponentData(iComp).Misc1,
"1")
  End If
'Module A parameters
  If ComponentData(iComp).ModuleA = "AV" Then
    Call CheckAVRoutine(iComp, ComponentData(iComp).DistribA, ComponentData(iComp).MiscA,
ComponentData(iComp).ManualVariability)
  ElseIf Len(Trim(ComponentData(iComp).ModuleA)) > 0 Then
    Call CheckRoutine(iComp, ComponentData(iComp).ModuleA, ComponentData(iComp).DistribA, ComponentData(iComp).MiscA,
"A")
  End If
'Module F parameters
  If ComponentData(iComp).ModuleF = "AV" Then
```

```
    Call CheckAVRoutine(iComp, ComponentData(iComp).DistribF, ComponentData(iComp).MiscF,
ComponentData(iComp).ManualVariability)
  ElseIf Len(Trim(ComponentData(iComp).ModuleF)) > 0 Then
    Call CheckRoutine(iComp, ComponentData(iComp).ModuleF, ComponentData(iComp).DistribF, ComponentData(iComp).MiscF,
"F")
  End If
'Module B parameters
  If ComponentData(iComp).ModuleB = "AV" Then
    Call CheckAVRoutine(iComp, ComponentData(iComp).DistribB, ComponentData(iComp).MiscB,
ComponentData(iComp).ManualVariability)
  ElseIf Len(Trim(ComponentData(iComp).ModuleB)) > 0 Then
    Call CheckRoutine(iComp, ComponentData(iComp).ModuleB, ComponentData(iComp).DistribB, ComponentData(iComp).MiscB,
"B")
  End If
Next

'Distribution Key loop
For iDK = 1 To DKMax
'If DK Master indicates Addends, it should have an entry from DK Addends
  If DKData(iDK).DKAddends Then
    DKFlag = False
    For iDetail = 1 To DetailMax
      If DKDetail(iDetail).DKNumber = DKData(iDK).DKComp Then
        DKFlag = True
        Exit For
      End If
    Next
    If Not DKFlag Then
      ErrDat.Book = CntlBook
      ErrDat.Sheet = CntlDKSheet
      ErrDat.Row = iDK + 1
      ErrDat.Level = 2
      ErrDat.Desc = "Distribution Key " & DKData(iDK).DKComp & " does not have an entry on the Addends sheet"
      Call RecordError
    End If
  End If
'Each DK on DK Master must have at least one component referencing it from the right module(s)
  If DKData(iDK).DKA Then
    Call CheckDKA(iDK)
  End If
  If DKData(iDK).DKF Then
    Call CheckDKF(iDK)
```

```
  End If
  If DKData(iDK).DKB Then
    Call CheckDKB(iDK)
  End If
Next

T1 = Timer
dT = T1 - T0
Debug.Print "Sub DataCheck execution time: " & Format(dT, "#,##0.000") & " seconds."
End Sub

Sub NonCompArg(iC, Desc)
'Prepares and records the error for an argument that is not a component or class
'iC is the component index
'Desc is a text description of the invalid argument
ErrDat.Level = 2
ErrDat.Book = CntlBook
ErrDat.Sheet = CntlCompSheet
ErrDat.Row = ComponentData(iC).Row
ErrDat.Desc = Desc & " for component " & ComponentData(iC).Label & " is not valid."
Call RecordError
End Sub
```

```
Public Sub CheckRoutine(iComp, RType, Arg1, Arg2, Modul)
'Checks that the Arguments for the routine are valid
'iComp is the component number (its place in the ComponentData array)
'RType is the Routine Type, e.g., AV, DA, DV, etc.
'Arg1 and Arg2 are the arguments (usually distribution keys) of the routine

Dim CheckFlag As Boolean
Dim ELevel As Integer
Dim EMsg As String
Dim SS0 As String
Dim SS1 As String
Dim SS2 As String

CheckFlag = False
EMsg = "iComp=" & iComp & "/RType=" & RType & "/Arg1=" & Arg1 & "/Arg2=" & Arg2 & "/Modul=" & Modul
ELevel = -1
RType = Trim(RType)
Modul = Trim(Modul)

Select Case RType
  Case "DA", "DV", "CV"
    If IsDK(Arg1) Then
      If Arg2 = 0 Then
        CheckFlag = True
      Else
        EMsg = "(" & RType & ") " & "Argument 2 (" & Arg2 & ") should be empty"
      End If
    Else
      EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") is not a valid Distribution Key"
    End If
  Case "DZ"
    If IsDK(Arg1) Then
      If Is983(Arg2) Then
        CheckFlag = True
      Else
        EMsg = "(" & RType & ") " & "Argument 2 (" & Arg2 & ") is not a valid CS98.3 component"
      End If
    Else
      EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") is not a valid Distribution Key"
    End If
  Case "DX"
    If IsDK(Arg1) Then
```

```
    If Is21983(Arg2) Then
      CheckFlag = True
    Else
      EMsg = "(" & RType & ") " & "Argument 2 (" & Arg2 & ") is not a valid CS21 or CS98.3 component"
    End If
  Else
    EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") is not a valid Distribution Key"
  End If
Case "CP"
  If IsComponent(Arg1) Then
    If IsComponent(Arg2) Then
      SS0 = ComponentData(iComp).SubSegment
      SS1 = ComponentData(CompIndx(Val(Arg1))).SubSegment
      SS2 = ComponentData(CompIndx(Val(Arg2))).SubSegment
      If SS0 = SS1 Then
        If SS0 = SS2 Then
          If Arg1 <> Arg2 Then
            CheckFlag = True
          Else
            EMsg = "(" & RType & ") " & "Arguments 1 and 2 must be different components"
          End If
        Else
          EMsg = "(" & RType & ") " & "Argument 2 (" & Arg2 & ") does not have the same subsegment as the component"
        End If
      Else
        EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") does not have the same subsegment as the component"
      End If
    Else
      EMsg = "(" & RType & ") " & "Argument 2 (" & Arg2 & ") is not a valid component"
    End If
  Else
    EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") is not a valid component"
  End If
Case "ST"
  If IsComponent(Arg1) Then
    If Arg2 = 0 Then
      SS0 = ComponentData(iComp).SubSegment
      SS1 = ComponentData(CompIndx(Val(Arg1))).SubSegment
      If SS0 = SS1 Then
        CheckFlag = True
      Else
        EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") does not have the same subsegment as the component"
```

```
      End If
    Else
      EMsg = "(" & RType & ") " & "Argument 2 (" & Arg2 & ") should be empty"
    End If
  Else
    EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") is not a valid component"
  End If
Case "SA"
  If IsComponent(Arg1) Then
    If Arg2 = 0 Then
      If Modul = "A" Then
        If Not ComponentData(CompIndx(Val(Arg1))).SubTotalA Then
          CheckFlag = True
        Else
          EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") is not a non-subtotal component"
          ELevel = 0
        End If
      ElseIf Modul = "F" Then
        If Not ComponentData(CompIndx(Val(Arg1))).SubTotalF Then
          CheckFlag = True
        Else
          EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") is not a non-subtotal component"
          ELevel = 0
        End If
      ElseIf Modul = "B" Then
        If Not ComponentData(CompIndx(Val(Arg1))).SubTotalB Then
          CheckFlag = True
        Else
          EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") is not a non-subtotal component"
          ELevel = 0
        End If
      ElseIf Modul = "C" Then
        If Not ComponentData(CompIndx(Val(Arg1))).SubTotalC Then
          CheckFlag = True
        Else
          EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") is not a non-subtotal component"
          ELevel = 0
        End If
      End If
    Else
      EMsg = "(" & RType & ") " & "Argument 2 (" & Arg2 & ") should be empty"
    End If
```

```
    Else
      EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") is not a valid component"
    End If
  Case "AJ"
    If Is983(Arg1) Then
      If IsClass(Arg2) Then
        CheckFlag = True
      Else
        EMsg = "(" & RType & ") " & "Argument 2 (" & Arg2 & ") is not a valid class"
      End If
    Else
      EMsg = "(" & RType & ") " & "Argument 1 (" & Arg1 & ") is not a valid component in CS98.3"
    End If
End Select

If CheckFlag Then
  Exit Sub
End If

ErrDat.Book = CntlBook
ErrDat.Sheet = CntlCompSheet
ErrDat.Row = ComponentData(iComp).Row
If ELevel < 0 Then
  ErrDat.Level = 3
Else
  ErrDat.Level = ELevel
End If
ErrDat.Desc = "Module " & Modul & ": " & EMsg
Call RecordError

End Sub
```

```
Public Sub CheckAVRoutine(iComp, Arg1, Arg2, Var)
'Checks the validity of input parameters when the AV routine has been specified

Dim EMsg As String

EMsg = ""

If Arg1 > 0 Then
  EMsg = "Argument 1 (" & Arg1 & ") should be empty"
  ErrDat.Level = 2
End If
If Arg2 > 0 Then
  EMsg = "Argument 2 (" & Arg2 & ") should be empty"
  ErrDat.Level = 2
End If
If Var <= 0 Then
  EMsg = "Manual variability (" & Var & ") expected to be greater than zero"
  ErrDat.Level = 3
End If

If EMsg = "" Then
  Exit Sub
End If

ErrDat.Book = CntlBook
ErrDat.Sheet = CntlCompSheet
ErrDat.Row = ComponentData(iComp).Row
ErrDat.Desc = "Module " & Modul & ": " & EMsg
Call RecordError

End Sub
```

```
Public Function Is983(C) As Boolean
'The argument, C, is a variant.
'If C is numeric and is the value of a valid component,
'and the argument is in CS 98.3, then the function returns True.
'Otherwise it returns False

Dim Flag983 As Boolean
Dim I As Integer

Is983 = False
Flag983 = False

If Not IsNumeric(C) Then
  Exit Function
End If

For I = 1 To CompsMax
  If C = Val(ComponentData(I).Label) Then
    Flag983 = True
    Debug.Print "Found component " & C
    Exit For
  End If
Next

If Not Flag983 Then
  Exit Function
End If

If ComponentData(CompIndx(C)).CS = 98.3 Then
  Is983 = True
Else
  Debug.Print "Component " & C & " CS = " & ComponentData(CompIndx(C)).CS
End If

End Function
```

```
Public Function Is21983(C) As Boolean
'The argument, C, is a variant.
'If C is numeric and is the value of a valid component,
'and the argument is in CS 21 or CS 98.3, then the function returns True.
'Otherwise it returns False

Dim Flag21983 As Boolean
Dim I As Integer

Is21983 = False
Flag21983 = False

If Not IsNumeric(C) Then
  Exit Function
End If

For I = 1 To CompsMax
  If C = Val(ComponentData(I).Label) Then
    Flag21983 = True
    Exit For
  End If
Next

If Not Flag21983 Then
  Exit Function
End If

If (ComponentData(CompIndx(C)).CS = 98.3) Or (ComponentData(CompIndx(C)).CS = 21) Then
  Is21983 = True
End If

End Function
```

```
Public Function IsDK(C) As Boolean
'The argument, C, is a variant.
'If C is numeric and is the value of a valid Distribution Key, the function returns True.
'Otherwise it returns False

Dim I As Integer

IsDK = False

If Not IsNumeric(C) Then
  Exit Function
End If

For I = 1 To DKMax
  If C = DKData(I).DKComp Then
    IsDK = True
    Exit For
  End If
Next

End Function
```

```
Public Sub CheckDKA(iDK)
'Checks to see whether the DK represented by iDK is referenced by at least one component in Module A

Dim Cp As Integer
Dim I As Integer

Cp = DKData(iDK).DKComp

For I = 1 To CompsMax
  If Cp = ComponentData(I).DistribA Or Cp = ComponentData(I).MiscA Then
    Exit Sub
  End If
Next

ErrDat.Book = CntlBook
ErrDat.Sheet = CntlDKSheet
ErrDat.Row = iDK + 1
ErrDat.Level = 0
ErrDat.Desc = "DK " & Cp & " is not referenced for Module A by any component."
Call RecordError

End Sub
```

```
Public Sub CheckDKF(iDK)
'Checks to see whether the DK represented by iDK is referenced by at least one component in Module F

Dim Cp As Integer
Dim I As Integer

Cp = DKData(iDK).DKComp

For I = 1 To CompsMax
  If Cp = ComponentData(I).DistribF Or Cp = ComponentData(I).MiscF Then
    Exit Sub
  End If
Next

ErrDat.Book = CntlBook
ErrDat.Sheet = CntlDKSheet
ErrDat.Row = iDK + 1
ErrDat.Level = 0
ErrDat.Desc = "DK " & Cp & " is not referenced for Module F by any component."
Call RecordError

End Sub
```

```
Public Sub CheckDKB(iDK)
'Checks to see whether the DK represented by iDK is referenced by at least one component in Module B

Dim Cp As Integer
Dim I As Integer

Cp = DKData(iDK).DKComp

For I = 1 To CompsMax
  If Cp = ComponentData(I).DistribB Or Cp = ComponentData(I).MiscB Then
    Exit Sub
  End If
Next

ErrDat.Book = CntlBook
ErrDat.Sheet = CntlDKSheet
ErrDat.Row = iDK + 1
ErrDat.Level = 0
ErrDat.Desc = "DK " & Cp & " is not referenced for Module B by any component."
Call RecordError

End Sub
```

XII. Directory of Global Variables

Public Variable Name	Description	Data Type	Declared	Where set	Where used	Data Source (If Applicable)
AddendMax	Unused					
AHeader1	A Report Header	String	ModReportVars	RptInit	Reports	Reports worksheet
AHeader2	A Report Header	String	ModReportVars	RptInit	Reports	Reports worksheet
BHeader1	B Report Header	String	ModReportVars	RptInit	Reports	Reports worksheet
BHeader2	B Report Header	String	ModReportVars	RptInit	Reports	Reports worksheet
CalcKeysCS	Cost segment for calculated distribution keys	Single	ModReportVars	RptInit	Reports	Reports worksheet
CHeader1	C Report Header	String	ModReportVars	RptInit	Reports	Reports worksheet
CHeader2	C Report Header	String	ModReportVars	RptInit	Reports	Reports worksheet
ClassFormulas()	Array to hold formulas for class subtotals	String array	ModPublicVariables	ClassSubtotals	RoutineAV RoutineDA RoutineDV RoutineDZ RoutineDX RoutineCV RoutineCP RoutineST RoutineSA RoutineAJ CalcDK RoutineAA RoutineAD	Internal
ClassIndx()	Array whose members point to the MailClassData array. The value of ClassIndx tells which element of MailClassData contains the Mail Class of interest, e.g., if you want data for Mail Class 109, use MailClassData(ClassIndx(109))	Integer array	ModPublicVariables	ClassArrayInit	UpdateOutputMatrix	Derived from MailClasses worksheet
ClassLabelRow	The row that contains labels for the Mail Class Columns	Long	ModPublicVariables	SetRowColVariables	SetRowColVariables WriteClasses	Internal
ClassMax	Actual number of mail classes in use	Integer	ModPublicVariables	Init	Init ClassArrayInit ReadInputMatrix MakeInputMatrixSheet ClassSubtotals GetREOptions	Derived from MailClasses worksheet
ClassNameCol	The column that contains Mail Class Names	Integer	ModPublicVariables	SetRowColVariables	WriteClasses	Internal
ClassNoCol	The column that contains Mail Class Numbers	Integer	ModPublicVariables	SetRowColVariables	WriteClasses	Internal
ClassRow()	Array to hold row for each mail class on worksheets	Integer array	ModPublicVariables	ClassArrayInit	RoutineAV RoutineDA RoutineDV RoutineDZ RoutineDX RoutineCV RoutineCP RoutineST RoutineSA RoutineAJ CalcDK RoutineAA RoutineAD RoutinePS UpdateOutputMatrix ClassSubtotals ComponentSubtotals	Derived from application of data in the MailClass array to the first Cost Segment sheet created.

XII. Directory of Global Variables

Public Variable Name	Description	Data Type	Declared	Where set	Where used	Data Source (If Applicable)
CntlBook	Name of the Control workbook	String	ModPublicVariables	ReadInputTableColumns RFCntlFileBrowse	CheckPhase2 ReadDistribution ReadDistKeys ReadDKDocumentation ReadDRptData DataCheck NonCompArg CheckRoutine CheckAVRoutine CheckDKA CheckDKF	InputTable sheet
CntlCompSheet	Name of the worksheet in the Control workbook that contains Component data	String	ModPublicVariables	ReadInputTableColumns	CheckPhase2 ReadDistribution DataCheck NonCompArg CheckRoutine CheckAVRoutine	InputTable sheet
CntlDKDetailSheet	Name of the worksheet in the Control workbook that contains Distribution Key detail (addends) data	String	ModPublicVariables	ReadInputTableColumns	CheckPhase2 ReadDKDocumentation	InputTable sheet
CntlDKSheet	Name of the worksheet in the Control workbook that contains Distribution Key data	String	ModPublicVariables	ReadInputTableColumns	CheckPhase2 ReadDistKeys DataCheck CheckDKA CheckDKF CheckDKB	InputTable sheet
CntlDReportSheet	Name of the sheet containing D Report data	String	ModPublicVariables	ReadInputTableColumns	ReadDRptData	InputTable sheet
ColHeadCol	The column that contains the headers for the column header	Integer	ModPublicVariables	SetRowColVariables	CSSheetCommonHeaders	Internal
CompColumn()	Indicates which column contains the component in the output workbook. This array is addressed directly, not through CompIndx	Integer array	ModPublicVariables	Init	RoutineAV RoutineDA RoutineDV RoutineDZ RoutineDX RoutineCV RoutineCP RoutineST RoutineSA RoutineAJ CalcDK RoutineAA RoutineAD RoutinePS WriteComps UpdateOutputMatrix FillDKSheet ComponentSubtotals	InputTable sheet
CompIndx()	Array whose members point to the ComponentData array. The value of CompIndx tells which element of ComponentData contains the Component of interest, e.g., if you want data for Component 601, use ComponentData(CompIndx(601))	Integer array	ModPublicVariables	ReadDistribution	DataCheck CheckRoutine Function Is983 Function Is21983 Reports KReport DReport	Derived from Comp Master worksheet in the Control workbook
CompNameRow	The row that contains the Component Names	Long	ModPublicVariables	SetRowColVariables	RoutineAJ WriteComps CSSheetCommonHeaders FillDKSheet	Internal

XII. Directory of Global Variables

Public Variable Name	Description	Data Type	Declared	Where set	Where used	Data Source (If Applicable)
CompNoRow	The row that contains the Component Numbers	Long	ModPublicVariables	SetRowColVariables	WriteComps CSSheetCommonHeaders FillDKSheet	Internal
ComponentData()	Contains basic information on all the components in the order in which the components appear on the 'Comp Master' worksheet in the Control workbook.	CompType array (User-defined data type)	ModPublicVariables	Init ReadDistribution InitExtraSegments	Function IsComponent DataCheck NonCompArg CheckRoutine CheckAVRoutine Function Is983 Function Is21983 CheckDKA CheckDKF CheckDKB CheckDKRF Phase1 Phase2 Phase3 Phase4 WriteComps FillDKSheet ComponentSubtotals Reports	Most data comes from the 'Comp Master' worksheet in the Control workbook
CompSheet()	Indicates the worksheet that contains the component. This array is addressed directly, not through CompIndx.	Integer array	ModPublicVariables	Init	RoutineAV RoutineDA RoutineDV RoutineDZ RoutineDX RoutineCV RoutineCP RoutineST RoutineSA RoutineAJ CalcDK RoutineAA RoutineAD RoutinePS WriteComps UpdateOutputMatrix FillDKSheet ComponentSubtotals	InputTable sheet

XII. Directory of Global Variables

Public Variable Name	Description	Data Type	Declared	Where set	Where used	Data Source (If Applicable)
CompsMax	Actual number of Components in use	Integer	ModPublicVariables	ReadDistribution	ReadDistribution InitExtraSegments IsComponent DataCheck Is983 Is21983 CheckDKA CheckDKF CheckDKB Phase1 Phase2 Phase3 Phase4 WriteComps FillDKSheet ComponentSubtotals	Derived from Comp Master worksheet in the Control workbook
Contingency	(RF support)	String	ModPublicVariables	EndRFOptions	unused	
ControlSheet	Name of the worksheet (in the CRA workbook) that contains the layout of the Control file	String	ModPublicVariables	Run	unused	Internal
CostMatrix()	Array of costs by Mail Class and Component	Double array	ModPublicVariables	Init ReadInputMatrix UpdateOutputMatrix GetRFOptions	UpdateOutputMatrix MakeInputMatrixSheet Reports	Input (I) File, but updated internally
CostSegment	Cost Segment names and numbers	CSType array (User-defined type)	ModPublicVariables	ReadDistribution	AddCSSheets Reports	Comp Master worksheet in the Control workbook
CRABook	Name of the CRA workbook	String	ModPublicVariables	Auto_Open	ReadLoaderModelData CheckPhase2 LoaderLayout DataCheck Driver RptInit Reports KReport KReport	Internal
CRAPath	Path (location) of the CRA workbook	String	ModPublicVariables	Auto_Open	Auto_Open ClassArrayInit RevArrayInit OpenWB	Internal
CSCount	Number of Cost Segments	Integer	ModPublicVariables	ReadDistribution	ReadDistribution	Derived from Comp Master worksheet in the Control workbook
CSIndex()	Array whose members point to the CostSegment array. The value of CSIndex tells which element of CostSegment contains the Cost Segment of interest, e.g., if you want data for C/S 9, use CostSegment(CSIndex(9))	Integer	ModPublicVariables	ReadDistribution	Reports	Derived from Comp Master worksheet in the Control workbook
CSMax	Actual number of Cost Segments in use	Integer	ModPublicVariables	ReadDistribution	AddCSSheets	Derived from Comp Master worksheet in the Control workbook
DCompCol	Column of sheet 'DReport' that contains component numbe	Integer	ModPublicVariables	ReadInputTableColumns	ReadDRptData	InputTable sheet
DData	Data for D report	DType array (User-defined type)	ModPublicVariables	ReadDRptData	RunDRoutines DReport	Sheet DReport in Control workbook
DetailMax	Maximum number of Distribution Key definitions allowed	Integer	ModPublicVariables	ReadDKDocumentation	DataCheck CalcDK	Derived from DK Addends sheet in Control workbook
DHeader1	D Report Header	String	ModReportVars	RptInit	DReport	Reports worksheet
DHeader2	D Report Header	String	ModReportVars	RptInit	DReport	Reports worksheet

XII. Directory of Global Variables

Public Variable Name	Description	Data Type	Declared	Where set	Where used	Data Source (If Applicable)
Distrib1Col	Column of 'Comp Master' that contains data to read into Componentdata.Distrib1	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
DistribACol	Column of 'Comp Master' that contains data to read into Componentdata.DistribA	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
DistribBCol	Column of 'Comp Master' that contains data to read into Componentdata.DistribB	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
DistribFCol	Column of 'Comp Master' that contains data to read into Componentdata.DistribF	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
DKAAddendCountRow	Row on 'DK Addends' which contains the count of the number of addends.	Long	ModPublicVariables	ReadInputTableColumns	ReadDKDocumentation	InputTable sheet
DKACol	Column of 'DK Master' that contains data to read into DKData.DKA	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistKeys	InputTable sheet
DKACompCol	Column of 'DK Addends' that contains data to read into DKDetail.DKANumber	Integer	ModPublicVariables	ReadInputTableColumns	ReadDKDocumentation	InputTable sheet
DKADataCol	First data column on 'DK Addends'. Data continues indefinitely to the right	Integer	ModPublicVariables	ReadInputTableColumns	ReadDKDocumentation	InputTable sheet
DKAddendsCol	Column of 'DK Master' that contains data to read into DKData.DKAddends	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistKeys	InputTable sheet
DKAffectedCol	Column of 'DK Master' that contains data to read into DKData.DKAffected	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistKeys	InputTable sheet
DKAFirstDataRow	First row of data on 'DK Addends'	Long	ModPublicVariables	ReadInputTableColumns	ReadDKDocumentation	InputTable sheet
DKAKeyRow	Row on 'DK Addends' which contains the Distribution Key numbe	Long	ModPublicVariables	ReadInputTableColumns	ReadDKDocumentation	InputTable sheet
DKATitleCol	Column of 'DK Addends' that contains data to read into DKDetail.DKATitle	Integer	ModPublicVariables	ReadInputTableColumns	ReadDKDocumentation	InputTable sheet
DKBCol	Column of 'DK Master' that contains data to read into DKData.DKB	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistKeys	InputTable sheet
DKCompCol	Column of 'DK Master' that contains data to read into DKData.DKComp	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistKeys	InputTable sheet
DKContentsCol	Column of 'DK Master' that contains data to read into DKData.DKContents	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistKeys	InputTable sheet
DKData	Contains basic information on all the Distribution Keys in the order in which the Distribution Keys appear on the 'DK Master' worksheet in the Control workbook	DKType array (User-defined data type)	ModPublicVariables	ReadDistKeys	DataCheck Function IsDK CheckDKA CheckDKF CheckDKB Phase2	DK Master worksheet in the Control workbook
DKDetail	Contains the list of components that are added together to make certain Distribution Keys, and contains related informatior	DKAddendType array (User-defined type)	ModPublicVariables	ReadDKDocumentation	DataCheck CalcDK	DK Addends worksheet in the Control workbook
DKFCol	Column of 'DK Master' that contains data to read into DKData.DKI	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistKeys	InputTable sheet
DKIndx()	Array whose members point to the DKtData array. The value of DKIndx tells which element of DKData contains the Distribution Key of interest, e.g., if you want data for Distribution Key component 1289 use ComponentData(CompIndx(1289))	Integer array	ModPublicVariables	ReadDistKeys	unused	Derived from DK Master worksheet in the Control workbook
DKMax	Actual number of Distribution Keys in use	Integer	ModPublicVariables	ReadDistKeys	DataCheck IsDK	Derived from DK Master worksheet in the Control workbook
DKRFCol	Column of 'DK Master' that contains data to read into DKData.DKRF	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistKeys	InputTable sheet
DKTitleCol	Column of 'DK Master' that contains data to read into DKData.DKTitle	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistKeys	InputTable sheet
DMax	Actual maximum number of components in D Report	Integer	ModPublicVariables	ReadDRptData	RunDRoutines DReport	Derived from sheet DReport
DocRow	The row that contains the Component Documentation	Long	ModPublicVariables	SetRowColVariables	WriteComps CSSheetCommonHeaders FillDKSheet	Internal
DParm1Col	Column of sheet 'DReport' that contains the first paramete	Integer	ModPublicVariables	ReadInputTableColumns	ReadDRptData	InputTable sheet
DParm2Col	Column of sheet 'DReport' that contains the second paramete	Integer	ModPublicVariables	ReadInputTableColumns	ReadDRptData	InputTable sheet
DParm3Col	Column of sheet 'DReport' that contains the third paramete	Integer	ModPublicVariables	ReadInputTableColumns	ReadDRptData	InputTable sheet

XII. Directory of Global Variables

Public Variable Name	Description	Data Type	Declared	Where set	Where used	Data Source (If Applicable)
DPrintCol	Column of sheet 'DReport' that contains indication whether to print in the D Report	Integer	ModPublicVariables	ReadInputTableColumns	ReadDRptData	InputTable sheet
DRFOnlyCol	Column of sheet 'DReport' that contains indication whether this component is only for RF run	Integer	ModPublicVariables	ReadInputTableColumns	ReadDRptData	InputTable sheet
DRoutineCol	Column of sheet 'DReport' that contains routine to use	Integer	ModPublicVariables	ReadInputTableColumns	ReadDRptData	InputTable sheet
DRpt	(RF support)	Boolean	ModPublicVariables	Reports DReport	ColFmtBorders DReport	Internal
ErrDat	Error data to be written on the Errors worksheet	ErrType (User-defined type)	ModPublicVariables	Auto_Open CheckPhase1 CheckPhase2 ReadDistKeys ReadDKDocumentation RecordError DataCheck NonCompArg CheckRoutine CheckAVRoutine CheckDKA CheckDKF CheckDKB RptInit	Auto_Open CheckPhase1 RecordError Driver	Internal
ErrMax	Maximum error level actually encountered in this run	Integer	ModPublicVariables	RecordError InitializeErrorSheet	Init Driver Run RunCleanup	Internal
ErrMaxAllow	User-specified maximum error level allowable for this run	Integer	ModPublicVariables	Init	Driver RunCleanup	Menu sheet of CRA Model
ErrRow	Next available row on the Errors worksheet	Long	ModPublicVariables	InitializeErrorSheet RecordError Run	RecordError RunCleanup	Internal
ErrSheet	Name of the worksheet in the CRA workbook on which errors are recorded	String	ModPublicVariables	Run	unused	Internal
ErrStop	Indicator whether this run has been halted because of errors	Boolean	ModPublicVariables	InitializeErrorSheet Driver	RunCleanup	Internal
FHeader1	F Report Header	String	ModReportVars	RptInit	Reports	Reports worksheet
FHeader2	F Report Header	String	ModReportVars	RptInit	Reports	Reports worksheet
FirstColumn	The first column that will have actual data	Integer	ModPublicVariables	SetRowColVariables CalcDK	RoutineAJ WriteComps FillDKSheet	Internal
FirstRow	The first row that will have actual data	Long	ModPublicVariables	SetRowColVariables	WriteClasses WriteComps FillDKSheet ClassArrayInit	Internal
FormResult	(RF support)	Integer	ModPublicVariables	GetRFOptions CancelForm CancelRFOptions	GetRFOptions	Rollforward input
HeadRow	The row that contains the worksheet heade	Long	ModPublicVariables	SetRowColVariables	ReadDRptData	Internal
IDecimals()	Parsed components to display in decimals	Integer array	ModReportVars	RptInit	Reports	Internal
IFileSheetName	Name of the worksheet expected in the "I" file workbook	String	ModPublicVariables	ReadInputTableColumns	CheckPhase2	InputTable sheet
IHeader1	Input Report Header	String	ModReportVars	RptInit	Reports	Reports worksheet
IHeader2	Input Report Header	String		RptInit	Reports	Reports worksheet

XII. Directory of Global Variables

Public Variable Name	Description	Data Type	Declared	Where set	Where used	Data Source (If Applicable)
iModule	Which module (numeric) we are currently running	Integer	ModPublicVariables	Driver	RoutineDriver Phase1 Phase2 Phase3 Phase4 WriteComps UpdateOutputMatrix AddCSSheets MakeWorkingBook FillDKSheet ComponentSubtotals Reports RptClasses ColFmtBorders	Internal
InputTypeCol	Column of 'Comp Master' that contains data to read into Componentdata.InputType	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
iPhase	Which of the (5) phases of routines is being processed	Integer	ModPublicVariables	unused	unused	Internal
IPrintDecimals	Components to display decimals in I report	String	ModReportVars	RptInit	RptInit	Reports worksheet
IRevComp	Components to display revenue classes in I report	Integer	ModReportVars	RptInit	Reports	Reports worksheet
IRptOK	Indicates whether any errors in parsing I report decimal to display string	Boolean	ModReportVars	RptInit	unused	Internal
iRunType	Type of run: 1=data check only / 2=Full Run / 3=Roll Forward	Integer	ModPublicVariables	ChangeRunType	Init CheckPhase2 ReadInputTableColumns Driver RptInit Reports DReport ChangeRunType	Menu sheet of CRA Model
IStartPrintRows	Input Report starting cost segment to print in rows rather than columns	Single		RptInit	Reports	Reports worksheet
KComps	Components to display in the K report	String	ModReportVars	RptInit	RptInit	Reports worksheet
KHeader1	K Report Header	String	ModReportVars	RptInit	KReport	Reports worksheet
KHeader2	K Report Header	String	ModReportVars	RptInit	KReport	Reports worksheet
KHeader3	K Report Header	String	ModReportVars	RptInit	KReport	Reports worksheet
KRpt	(RF support)	Boolean	ModPublicVariables	Reports KReport	ColFmtBorders KReport	Internal
KRptComps	Array that KComps is parsed into	Integer	ModReportVars	RptInit	KReport	Internal
KRptOK	Flag to indicate whether K report can be run	Boolean	ModReportVars	RptInit	unused	Internal
LastCS	Last real cost segment. Remaining segments are distribution keys	Single	ModReportVars	Init RptInit	ReadDistribution InitExtraSegments ComponentSubtotals Reports	Limits sheet
LibRefCol	Column of 'Comp Master' that contains data to read into Componentdata.LibRef	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
LimitSheet	Name of the worksheet (in the CRA workbook) that contains limits	String	ModPublicVariables	Run	unused	Internal
ListSheet	Name of the worksheet (in the CRA workbook) that contains lists	String	ModPublicVariables	Run	unused	Internal
LoaderClassRange	Named range in the Loader where Mail Class data starts	String	ModPublicVariables	LoaderLayout	ClassArrayInit	InputTable sheet
LoaderCurrentDir	Named range in the Loader containing the path current mode	String	ModPublicVariables	LoaderLayout	unused	InputTable sheet
LoaderCurrentModel	Named range in the Loader containing the current mode	String	ModPublicVariables	LoaderLayout	unused	InputTable sheet
LoaderModelsRange	Named range immediately above the list of models in the loade	String	ModPublicVariables	LoaderLayout	ReadLoaderModelData	InputTable sheet
LoaderModelsSheet	Name of the worksheet containina ll the models available in the loader	String	ModPublicVariables	LoaderLayout	unused	InputTable sheet
LoaderName	Workbook name for the Data Entry Module	String	ModPublicVariables	LoaderLayout	ReadLoaderModelData ClassArrayInit RevArrayInit	Internal

XII. Directory of Global Variables

Public Variable Name	Description	Data Type	Declared	Where set	Where used	Data Source (If Applicable)
LoaderPath	Path (location) of the Data Entry Module	String	ModPublicVariables	Auto_Open	ReadLoaderModelData	Assumed to be the same as the CRA Model
LoaderRevRange	Range for revenue Mail Classes in Loader workbook	String	ModPublicVariables	LoaderLayout	RevArrayInit	InputTable sheet
MailClassData	Contains basic information on all the Mail Classes in the order in which the Mail Classes appear on the 'MailClass' worksheet	MailClassType array (User-defined data type	ModPublicVariables	ClassArrayInit	Function IsClass RoutineAV RoutineDA RoutineDV RoutineDZ RoutineDX RoutineCV RoutineCP RoutineSA RoutineAJ CalcDK RoutineAA RoutineAD RoutinePS WriteClasses WriteComps UpdateOutputMatrix FillDKSheet ClassSubtotals ComponentSubtotals RptClasses	MailClasses worksheet
ManualVariabilityCol	Column of 'Comp Master' that contains data to read into Componentdata.ManualVariability	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
MatrixColOffset	Offset from first column to start of data (in InputMatrix and OutputMatrix)	Integer	ModPublicVariables	SetRowColVariables	RoutineAV RoutineDA RoutineDV RoutineDZ RoutineDX RoutineCV RoutineAJ WriteComps UpdateOutputMatrix MakeInputMatrixSheet FillDKSheet GetRFOptions	Internal
MatrixRowOffset	Offset from first row to start of data (in InputMatrix and OutputMatrix)	Integer	ModPublicVariables	SetRowColVariables	RoutineAV RoutineDA RoutineDV RoutineDZ RoutineDX RoutineCV RoutineAJ WriteComps UpdateOutputMatrix MakeInputMatrixSheet FillDKSheet GetRFOptions	Internal

XII. Directory of Global Variables

Public Variable Name	Description	Data Type	Declared	Where set	Where used	Data Source (If Applicable)
MaxClass	Maximum number of Mail Classes (and highest Class Number) allowed	Integer	ModPublicVariables	Init	Init ClassArrayInit ReadInputMatrix MakeInputMatrixSheet ClassSubtotals InitClassRow	Limits sheet
MaxCompon	Maximum number of Components allowed in a Cost Segmen	Integer	ModPublicVariables	Init	Reports	Limits sheet
MaxComps	Maximum number of Components (and highest Component number) allowed	Integer	ModPublicVariables	Init	Init ReadDistribution ReadDistKeys ReadInputMatrix InitMtxChange UpdateOutputMatrix MakeInputMatrixSheet	Limits sheet
MaxCS	Maximum number of Cost Segments allowed	Integer	ModPublicVariables	Init	Unused	Limits sheet
MaxDefs	Maximum number of Distribution Key definitions allowed	Integer	ModPublicVariables	Init	ReadDistKeys Phase2	Limits sheet
MaxDKComp	Maximum number of components allowed in a Distribution Key	Integer	ModPublicVariables	Init	unused	Limits sheet
MenuSheet	Name of the worksheet (in the CRA workbook) that contains the menu	String	ModPublicVariables	Run	Run	Internal
Misc1Col	Column of 'Comp Master' that contains data to read into Componentdata.Misc1	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
MiscACol	Column of 'Comp Master' that contains data to read into Componentdata.MiscA	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
MiscBCol	Column of 'Comp Master' that contains data to read into Componentdata.MiscB	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
MiscFCol	Column of 'Comp Master' that contains data to read into Componentdata.MiscF	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
ModName	Name of the model being run	String	ModPublicVariables	ReadLoaderModelData ReadRunData SelectModel RFCntlFileBrowse	ReadLoaderModelData CheckPhase2 ReadInputTableColumns ReadInputMatrix MakeWorkingBook RptInit Reports KReport	Internal
ModPath	Path (location) of the model being run	String	ModPublicVariables	ReadLoaderModelData ReadRunData SelectModel GetRFOptions RFCntlFileBrowse	ReadLoaderModelData CheckPhase2 ReadDistribution ReadInputMatrix MakeWorkingBook Reports KReport	Internal
Module1Col	Column of 'Comp Master' that contains data to read into Componentdata.Module1	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
ModuleACol	Column of 'Comp Master' that contains data to read into Componentdata.ModuleA	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
ModuleBCol	Column of 'Comp Master' that contains data to read into Componentdata.ModuleB	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
ModuleFCol	Column of 'Comp Master' that contains data to read into Componentdata.ModuleF	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet

XII. Directory of Global Variables

Public Variable Name	Description	Data Type	Declared	Where set	Where used	Data Source (If Applicable)
ModuleLabel()	Array of labels for the modules: 1, A, F, B, C	String array	ModPublicVariables	SetRowColVariables	RoutineDriver UpdateOutputMatrix AddCSSheets MakeWorkingBook ComponentSubtotals Reports	Internal
MtxChange()	Each element of this array indicates whether the corresponding elements of CostMatrix have changed: False=not changed / True=Changed	Boolean array	ModPublicVariables	InitMtxChange Phase1 Phase2 Phase3 Phase4 RunDRoutines UpdateOutputMatrix ComponentSubtotals	UpdateOutputMatrix	Internal
NumberCol	Column of 'Comp Master' that contains data to read into Componentdata.Number	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
OtherClass	Class number for Other (Institutional) cost	Integer	ModPublicVariables	Init	RoutineAV RoutineDV RoutineDC RoutineDX RoutineCV RoutineCP RoutineST RunDRoutines RoutineAA	Limits sheet
ParentACol	Column of 'Comp Master' that contains data to read into Componentdata.ParentA	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
ParentBCol	Column of 'Comp Master' that contains data to read into Componentdata.ParentB	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
ParentCCol	Column of 'Comp Master' that contains data to read into Componentdata.ParentC	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
ParentFCol	Column of 'Comp Master' that contains data to read into Componentdata.ParentF	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
PlaceHolderCount	Number of Place-holder worksheets	Integer	ModPublicVariables	InitExtraSegments	AddCSSheets	Derived from Comp Master worksheet in the Control workbook
PlaceHolderSheets()	Names of worksheets that are similar to Cost Segments, but are for place-holders, e.g., 21, 98.1, etc.	String array	ModPublicVariables	InitExtraSegments	AddCSSheets	Derived from Comp Master worksheet in the Control workbook
PlaceHolderTypes()	Descriptions of the placeholder worksheets. These descriptions print out as part of the title	Integer array	ModPublicVariables	InitExtraSegments	AddCSSheets	Derived from Comp Master worksheet in the Control workbook
PrintACol	Column of 'Comp Master' that contains data to read into Componentdata.PrintA	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
PrintBCol	Column of 'Comp Master' that contains data to read into Componentdata.PrintB	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
PrintCCol	Column of 'Comp Master' that contains data to read into Componentdata.PrintC	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
PrintFCol	Column of 'Comp Master' that contains data to read into Componentdata.PrintF	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
PrintICol	Column of 'Comp Master' that contains data to read into Componentdata.PrintI	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
RateCase	(RF support)	String	ModPublicVariables	EndRFOptions	unused	
RevClassData	Similar to MailClassData, but holds information on Revenue classes	MailClassType array (User-defined data type	ModPublicVariables	RevArrayInit	RptRevClasses ColFmtBorders	RevenueClasses sheet in Loader module

XII. Directory of Global Variables

Public Variable Name	Description	Data Type	Declared	Where set	Where used	Data Source (If Applicable)
RevMax	Actual maximum number of classes in RevClassData	Integer	ModPublicVariables	RevArrayInit	RptRevClasses ColFmtBorders	Derived from RevenueClasses sheet in Loader module
RevSheet	Indication whether we are working on the Revenue Shee	Boolean	ModReportVars	Reports	ColFmtBorders	Internal
Rfwd	(RF support)	Integer	ModPublicVariables	EndRFOptions	RptInit	Internal
RFYear	(RF support)	String	ModPublicVariables	EndRFOptions	RptInit	Internal
RptCentury	Century to show on report	String	ModReportVars	RptInit	RptInit	Reports worksheet
RptClassRow	Class Row	Long array	ModReportVars	unused	unused	
RptCompRow	Component Row	Integer	ModReportVars	RptInit	Reports ColFmtBorders KReport DReport	Reports worksheet
RptStartCol	Starting column for data	Integer	ModReportVars	RptInit	Reports ReportClasses ReportRevClasses ColFmtBorders RowFmtBorders KReport DReport	Reports worksheet
RptStartRow	Starting row for data	Integer	ModReportVars	RptInit	Reports ReportClasses ReportRevClasses ColFmtBorders RowFmtBorders KReport DReport	Reports worksheet
RunOK	(RF support)	Boolean	ModPublicVariables	Driver GetRFOptions RFCntlFileBrowser	Driver	Internal
SaveCloseFlag()	Array indicating whether each individual file needs to be saved and closed	Boolean array	ModPublicVariables	Init OpenWB	SaveCloseWorkbooks	Internal
SaveCloseNames()	Names of the workbooks that need to be saved and closed	String array	ModPublicVariables	Init OpenWB	SaveCloseWorkbooks	Internal
SCCount	Number of workbooks that need to be saved and closed at the end of the run	Integer	ModPublicVariables	Init OpenWB	SaveCloseWorkbooks OpenWB	Internal
SCFlag	Set by calling routine, indicates whether a specific file needs to be saved and closed	Boolean	ModPublicVariables	ReadLoaderModelData ReadDistribution ReadInputMatrix OpenWB	OpenWB	Internal
SegCol	The column that contains the Row Sequence Numbe	Integer	ModPublicVariables	SetRowColVariables	WriteClasses	Internal
SkipClass	Class number to skip to when reading input data (skip everything below this number)	Integer	ModPublicVariables	Init	ReadInputMatrix GetRFOptions	Limits sheet
SourceCol	Column of 'Comp Master' that contains data to read into Componentdata.Source	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
SubSegmentCol	Column of 'Comp Master' that contains data to read into Componentdata.SubSegment	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
TitleCol	Column of 'Comp Master' that contains data to read into Componentdata.Title	Integer	ModPublicVariables	ReadInputTableColumns	ReadDistribution	InputTable sheet
TotClass	Class number for total accrued cost	Integer	ModPublicVariables	Init	RoutineDA RoutineDV RoutineDZ RoutineDX RoutineCP RoutineST Reports DReport	Limits sheet

XII. Directory of Global Variables

Public Variable Name	Description	Data Type	Declared	Where set	Where used	Data Source (If Applicable)
TotVVClass	Class number for Total Attributable (Volume Variable) cost	Integer	ModPublicVariables	Init	RoutineDV RoutineDZ RoutineDX RoutineCV RoutineCP RoutineAJ	Limits sheet
USPSorPRC	Indication whether this is a USPS or PRC run	Integer	ModReportVars	RptInit	RptInit Reports DReport	Menu sheet of CRA Model
VariabilityRow	The row that contains the Volume Variabilit	Long	ModPublicVariables	SetRowColVariables	unused	Internal
VerType	(RF support)	Integer	ModPublicVariables	EndRFOptions	RptInit	Internal
WorkAreaCol	Points to column on WorkArea worksheet	Integer	ModPublicVariables	RoutineDriver RoutineAJ	RoutineAJ	Internal
WorkingBook	Name of the workbook in which calculations are currently being done.	String	ModPublicVariables	MakeWorkingBook	AddCSSheets MakeWorkingBook MakeInputMatrixShee	Internal
xlMaxLine	Maximum number of lines on an Excel worksheet	Long	ModPublicVariables	ModPublicVariables	ReadDistribution ReadDistKeys ReadDKDocumentation InitializeErrorSheet Reports ColFmtBorders RowFmtBorders xlFmt	Internal
ynSaveClose	Indicates whether the user wants files saved and closed at the end of the run	Boolean	ModPublicVariables	Run	SaveCloseWorkbooks	Menu sheet of CRA Model

Definition of User Defined Data Types

Type	Elements	Element Type	Description of Elements
CompType	Label	String	Component Number expressed as alpha and padded with zeros, e.g., "0251" is Postmaste
	Title	String	Component Title, e.g., "C/S 1 Postmasters
	SubSegment	String	Hierarchical description, e.g., 3.3.1 is Administrative Clerk
	Source	String	Description of the source of this data elemer
	LibRef	String	Library Reference
	InputType	String	Takes the following values (only the first characters is acted upon) R=Rate / T=Total / U=USPS Total, PRC Rate / P=PRC Total, USPS Rate
	ManualVariability	Single	Variability Rate (read as a percentage
	PrintI	String	Which "I" reports this Component is printed in: Both, USPS, or PRC (only the first letter is rea
	Module1	String	Which routine is used for the Module1 (pre-A) calculatio
	Distrib1	Integer	Distribution Key or Source of Module1 informatio
	Misc1	String	Additional (optional) parameter for Module1 (required for certain values of Module
	ModuleA	Integer	Which routine is used for the ModuleA calculatio
	DistribA	Integer	Distribution Key or Source of ModuleA informatio
	MiscA	Integer	Additional (optional) parameter for ModuleF (required for certain values of Module
	PrintA	String	Which "A" reports this Component is printed in: Both, USPS, or PRC (only the first letter is rea
	ParentA	Integer	The number of the component that is the "Parent" of this component for Module
	ModuleF	String	Which routine is used for the ModuleF calculatio
	DistribF	Integer	Distribution Key or Source of ModuleF informatio
	MiscF	Integer	Additional (optional) parameter for ModuleF (required for certain values of Module
	PrintF	String	Which "F" reports this Component is printed in: Both, USPS, or PRC (only the first letter is rea
	ParentF	Integer	The number of the component that is the "Parent" of this component for Module
	ModuleB	String	Which routine is used for the ModuleB calculatio
	DistribB	Integer	Distribution Key or Source of ModuleB informatio
	MiscB	Integer	Additional (optional) parameter for ModuleB (required for certain values of Module
	PrintB	String	Which "B" reports this Component is printed in: Both, USPS, or PRC (only the first letter is rea
	ParentB	Integer	The number of the component that is the "Parent" of this component for Module
	PrintC	String	Which "C" reports this Component is printed in: Both, USPS, or PRC (only the first letter is rea
	ParentC	Integer	The number of the component that is the "Parent" of this component for Module
	CS	Single	Cost Segment to which this component belongs. This variable is derive
	SubTotalA	Boolean	Indicates whether this component is a SubTotal for purposes of ModuleA. This variable is derive
	SubTotalF	Boolean	Indicates whether this component is a SubTotal for purposes of ModuleF. This variable is derive
	SubTotalB	Boolean	Indicates whether this component is a SubTotal for purposes of ModuleB. This variable is derive
	SubTotalC	Boolean	Indicates whether this component is a SubTotal for purposes of ModuleC. This variable is derive
	Row	Long	The row in the control table that contains data for this component. This variable is derive

Definition of User Defined Data Types

Type	Elements	Element Type	Description of Elements
MailClassType	Name	String	Mail Class Name
	Number	Integer	Mail Class Number
	Level	Integer	Mail Class Leve
	Parent	Integer	Parent of this Mail Class (i.e., the class that is the subtotal containing this Mail Clas
	SubTotal	Boolean	Whether this Mail Class is a Subtotal class
	Display1	Integer	Indicator of how to display this Mail Clas
	Display2	Integer	Indicator of how to display this Mail Clas
	Border	Integer	Indicator of what type of border to use when displaying this Mail Cla
DKType	DKComp	Integer	Distribution Key component numbe
	DKTitle	String	Distribution Key Title
	DKContents	String	Description of the contents of the Distribution Ke
	DKAffected	String	Concatenated string of the components affected by this Distribution Ke
	DKAddends	Boolean	Indication of whether this DK has addend
	DKA	Boolean	Whether this DK is used in ModuleA
	DKF	Boolean	Whether this DK is used in ModuleF
	DKB	Boolean	Whether this DK is used in ModuleB
	DKRF	Boolean	Whether this DK is used in ModuleRF
DKAddendType	DKATitle	String	Distribution Key Title
	DKNumber	Integer	Distribution Key component numbe
	AddendCount	Integer	Number of addends
	Addends(100)	xInteger array	Array of addends. Each element of the array, up to the AddendCount, is a component numl
CSType	CSNumber	Integer	Cost Segment Number
	CSName	String	Cost Segment Name
ErrType	Level	Integer	Level (severity) of this error. 0=no error, information only / 1=error / 2=severe error / 3=fatal e
	Book	String	Workbook containing the data found to be in erro
	Sheet	String	Worksheet containing the data found to be in erro
	Row	Long	Row containing the data found to be in erro
	Desc	String	Description of the error
DType	DComp	Integer	Component Number
	DRoutine	String	Routine to execute
	DParm1	Integer	First routine paramete
	DParm2	Integer	Second routine paramete
	DParm3	Integer	Third routine paramete
	DPrint	String	Whether this component is printed in the D repor
	DRFOnly	String	Whether this line applies only to the Rollforward D Repo