
**Rochester Institute of Technology - Masters Degree Program
School of Computer Science**

THESIS FINAL REPORT

VOLUME II

**SUBJECT: AUTOMATED GENERATION OF SW
DESIGN CONSTRUCTS FROM MESA
SOURCE CODE**

AUTHOR: DAVID EGERTON

DATED: 4Q1993

REVISION: 1.0

**Rochester Institute of Technology - Masters Degree Program
School of Computer Science**

**Automated generation of SW design constructs
from Mesa source code:**

by

David Egerton

**A thesis submitted to the School of Computer Science
in partial fulfillment of the requirements of the degree of
Master of Science in Computer Science.**

**NOTE: VOLUME I CONTAINS THE MAIN THESIS,
VOLUME II THE SUPPORTING APPENDICES**

Rochester Institute of Technology - Masters Degree Program
School of Computer Science

TABLE OF CONTENTS

VOLUME II

APPENDICES

- Appendix A Output listings
 - A.1 Extractor.data
 - A.2 StructChart.data
 - A.2.1 SCGraph
 - A.2.2 SCTable
 - A.3 DataFlow.data
 - A.3.1 DFD Variable
 - A.3.2 DFD Table
- Appendix B Source code listings
 - B.1 Configuration file
 - B.2 Interface files
 - B.3 Program files
- Appendix C PGS Grammar
- Appendix D Variable types
- Appendix E Variable scopes
- Appendix F Repeats
- Appendix G List types

APPENDIX A - OUTPUT LISTINGS

A

OUTPUT LISTINGS

APPENDIX A - OUTPUT LISTINGS

A.1

OUTPUT LISTING OF EXTRACTOR.DATA

DesignExtractor 1.0 - By: David Egerton

Meta Source Files: DesignExtractorImpl

Parse! Display! Clear!

SC Graph

SC Table

OFD Variable

OFD Table

Raw Data

Repeats

Pass4 ... Count Procedure declarations

Pass5 ... Determine scope of variables ,done.

Cumulative Buffer Size = 3778 File Count = 1 ...

LOC this file = 378 ...

Analysis Complete. 1 file.

~ ~ ~ Processing Display ~ ~ ~ ~

Creating output file . .

Lines processed , , 0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100,
1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000, 2100, 2200, 2300, 2400,
2500, 2600, 2700, 2800, 2900, 3000, 3100, 3200, 3300, 3400, 3500, 3600, 3700,
Done , , Output created to file Extractor.data

~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

~ ~ ~ ~ Display Complete ~ ~ ~ ~ ~

DESIGN EXTRACTION STATISTICS
=====

The files parsed in this session were:

Index	File Name	Start Index	End Index
1	DesignExtractorImpl.mesa	1	3778

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	Id mark	Var mark	Var Scpe	Read Write	Decl Loca	Import Op	Import.....Export Identifier	Type	Values
1	DIRECTORY	64	1	0	0	64	0	0	-	-	0	-	-	-	-
2	ChangeTable	1	1	0	0	64	64	0	-	-	0	-	-	-	-
3	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
4	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
5	Add	1	1	0	0	69	0	0	-	-	0	-	-	-	-
6	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
7	AllEntryNames	1	1	0	0	69	0	0	-	-	0	-	-	-	-
8	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
9	CleanUp	1	1	0	0	69	0	0	-	-	0	-	-	-	-
10	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
11	Handle	1	1	0	0	69	0	0	-	-	0	-	-	-	-
12	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
13	Init	1	1	0	0	69	0	0	-	-	0	-	-	-	-
14	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
15	MakeTable	1	1	0	0	69	0	0	-	-	0	-	-	-	-
16]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
17	,	9	1	0	0	64	0	0	-	-	0	-	-	-	-
18	Display	1	1	0	0	64	64	0	-	-	0	-	-	-	-
19	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
20	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
21	Control	1	1	0	0	69	0	0	-	-	0	-	-	-	-
22]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
23	,	9	1	0	0	64	0	0	-	-	0	-	-	-	-
24	Environment	1	1	0	0	64	64	0	-	-	0	-	-	-	-
25	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
26	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
27	wordsPerPage	1	1	0	0	69	0	0	-	-	0	-	-	-	-
28]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
29	,	9	1	0	0	64	0	0	-	-	0	-	-	-	-
30	Exec	1	1	0	0	64	64	0	-	-	0	-	-	-	-
31	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
32	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
33	AddCommand	1	1	0	0	69	0	0	-	-	0	-	-	-	-
34	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
35	ExecProc	1	1	0	0	69	0	0	-	-	0	-	-	-	-
36	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
37	LookupCommand	1	1	0	0	69	0	0	-	-	0	-	-	-	-
38	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
39	OutputProc	1	1	0	0	69	0	0	-	-	0	-	-	-	-
40	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
41	RemoveCommand	1	1	0	0	69	0	0	-	-	0	-	-	-	-
42]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
43	,	9	1	0	0	64	0	0	-	-	0	-	-	-	-
44	Format	1	1	0	0	64	64	0	-	-	0	-	-	-	-
45	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
46	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
47	Line	1	1	0	0	69	0	0	-	-	0	-	-	-	-
48	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lv1	Proc lv1	List mark	id mark	Var mark	Var Scpe	Read Wrt	Decl Loca	Import OpExport Identifier	Type	Values
49	LongNumber	1	1	0	0	69	0	0	-	-	0	-	-	-	-
50	StringProc	9	1	0	0	69	0	0	-	-	0	-	-	-	-
51	StringProc	1	1	0	0	69	0	0	-	-	0	-	-	-	-
52	StringProc	9	1	0	0	69	0	0	-	-	0	-	-	-	-
53	Text	1	1	0	0	69	0	0	-	-	0	-	-	-	-
54	Text	139	1	0	0	64	0	0	-	-	0	-	-	-	-
55	FormSW	9	1	0	0	64	0	0	-	-	0	-	-	-	-
56	FormSW	1	1	0	0	64	64	0	-	-	0	-	-	-	-
57	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
58	USING	145	1	0	0	69	0	0	-	-	0	-	-	-	-
59	AllocateItemDescript	1	1	0	0	69	0	0	-	-	0	-	-	-	-
60	AllocateItemDescript	9	1	0	0	69	0	0	-	-	0	-	-	-	-
61	BooleanItem	1	1	0	0	69	0	0	-	-	0	-	-	-	-
62	BooleanItem	9	1	0	0	69	0	0	-	-	0	-	-	-	-
63	ClientItemsProcType	1	1	0	0	69	0	0	-	-	0	-	-	-	-
64	ClientItemsProcType	9	1	0	0	69	0	0	-	-	0	-	-	-	-
65	CommandItem	1	1	0	0	69	0	0	-	-	0	-	-	-	-
66	CommandItem	9	1	0	0	69	0	0	-	-	0	-	-	-	-
67	FreeHintsProcType	1	1	0	0	69	0	0	-	-	0	-	-	-	-
68	FreeHintsProcType	9	1	0	0	69	0	0	-	-	0	-	-	-	-
69	line0	1	1	0	0	69	0	0	-	-	0	-	-	-	-
70	line0	9	1	0	0	69	0	0	-	-	0	-	-	-	-
71	line1	1	1	0	0	69	0	0	-	-	0	-	-	-	-
72	line1	9	1	0	0	69	0	0	-	-	0	-	-	-	-
73	MenuProcType	1	1	0	0	69	0	0	-	-	0	-	-	-	-
74	MenuProcType	9	1	0	0	69	0	0	-	-	0	-	-	-	-
75	nextPlace	1	1	0	0	69	0	0	-	-	0	-	-	-	-
76	nextPlace	9	1	0	0	69	0	0	-	-	0	-	-	-	-
77	ProcType	1	1	0	0	69	0	0	-	-	0	-	-	-	-
78	ProcType	9	1	0	0	69	0	0	-	-	0	-	-	-	-
79	sameLine	1	1	0	0	69	0	0	-	-	0	-	-	-	-
80	sameLine	9	1	0	0	69	0	0	-	-	0	-	-	-	-
81	StringItem	1	1	0	0	69	0	0	-	-	0	-	-	-	-
82	StringItem	139	1	0	0	64	0	0	-	-	0	-	-	-	-
83	StringItem	9	1	0	0	64	0	0	-	-	0	-	-	-	-
84	Heap	1	1	0	0	64	64	0	-	-	0	-	-	-	-
85	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
86	USING	145	1	0	0	69	0	0	-	-	0	-	-	-	-
87	Create	1	1	0	0	69	0	0	-	-	0	-	-	-	-
88	Create	9	1	0	0	69	0	0	-	-	0	-	-	-	-
89	Delete	1	1	0	0	69	0	0	-	-	0	-	-	-	-
90	Delete	9	1	0	0	69	0	0	-	-	0	-	-	-	-
91	SystemZone	1	1	0	0	69	0	0	-	-	0	-	-	-	-
92	SystemZone	139	1	0	0	64	0	0	-	-	0	-	-	-	-
93	SystemZone	9	1	0	0	64	0	0	-	-	0	-	-	-	-
94	HeraldWindow	1	1	0	0	64	64	0	-	-	0	-	-	-	-
95	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
96	USING	145	1	0	0	69	0	0	-	-	0	-	-	-	-
97	AppendMessage	1	1	0	0	69	0	0	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS
=====

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scope	Read Write	Oec1 Loca	Import OpExport Identifier	Type	Values
98] MesaTab USING	139	1	0	0	64	0	0	-	-	0	-	-	-	-
99		9	1	0	0	64	0	0	-	-	0	-	-	-	-
100		1	1	0	0	64	64	0	-	-	0	-	-	-	-
101	[69	1	0	0	64	0	0	-	-	0	-	-	-	-
102		145	1	0	0	69	0	0	-	-	0	-	-	-	-
103		139	1	0	0	64	0	0	-	-	0	-	-	-	-
104	, MFile USING	9	1	0	0	64	0	0	-	-	0	-	-	-	-
105		1	1	0	0	64	64	0	-	-	0	-	-	-	-
106		69	1	0	0	64	0	0	-	-	0	-	-	-	-
107	[EnumerateDirectory	145	1	0	0	69	0	0	-	-	0	-	-	-	-
108		1	1	0	0	69	0	0	-	-	0	-	-	-	-
109		9	1	0	0	69	0	0	-	-	0	-	-	-	-
110	, EnumerateProc	1	1	0	0	69	0	0	-	-	0	-	-	-	-
111		9	1	0	0	69	0	0	-	-	0	-	-	-	-
112		1	1	0	0	69	0	0	-	-	0	-	-	-	-
113] Error	139	1	0	0	64	0	0	-	-	0	-	-	-	-
114		9	1	0	0	64	0	0	-	-	0	-	-	-	-
115		1	1	0	0	64	64	0	-	-	0	-	-	-	-
116	MSegment USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
117		145	1	0	0	69	0	0	-	-	0	-	-	-	-
118		1	1	0	0	69	0	0	-	-	0	-	-	-	-
119	, Create	9	1	0	0	69	0	0	-	-	0	-	-	-	-
120		1	1	0	0	69	0	0	-	-	0	-	-	-	-
121		9	1	0	0	69	0	0	-	-	0	-	-	-	-
122	Delete	1	1	0	0	69	0	0	-	-	0	-	-	-	-
123		9	1	0	0	69	0	0	-	-	0	-	-	-	-
124		1	1	0	0	69	0	0	-	-	0	-	-	-	-
125] Handle	139	1	0	0	64	0	0	-	-	0	-	-	-	-
126		9	1	0	0	64	0	0	-	-	0	-	-	-	-
127		1	1	0	0	64	64	0	-	-	0	-	-	-	-
128	MStream USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
129		145	1	0	0	69	0	0	-	-	0	-	-	-	-
130		1	1	0	0	69	0	0	-	-	0	-	-	-	-
131	Create	9	1	0	0	69	0	0	-	-	0	-	-	-	-
132		1	1	0	0	69	0	0	-	-	0	-	-	-	-
133		9	1	0	0	69	0	0	-	-	0	-	-	-	-
134	, Handle	1	1	0	0	69	0	0	-	-	0	-	-	-	-
135		139	1	0	0	64	0	0	-	-	0	-	-	-	-
136		9	1	0	0	64	0	0	-	-	0	-	-	-	-
137	ParseTable	1	1	0	0	64	64	0	-	-	0	-	-	-	-
138		9	1	0	0	64	0	0	-	-	0	-	-	-	-
139		1	1	0	0	64	64	0	-	-	0	-	-	-	-
140	Parser USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
141		145	1	0	0	69	0	0	-	-	0	-	-	-	-
142		1	1	0	0	69	0	0	-	-	0	-	-	-	-
143	AtomIndex	9	1	0	0	69	0	0	-	-	0	-	-	-	-
144		1	1	0	0	69	0	0	-	-	0	-	-	-	-
145		9	1	0	0	69	0	0	-	-	0	-	-	-	-
146	AtomProps	1	1	0	0	69	0	0	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Wrt	Decl Loca	Import OpExport Identifier	Type	Values
147	, AtomStack	9	1	0	0	69	0	0	-	-	0	-	-	-	-
148		1	1	0	0	69	0	0	-	-	0	-	-	-	-
149	, AtomStackPtr	9	1	0	0	69	0	0	-	-	0	-	-	-	-
150		1	1	0	0	69	0	0	-	-	0	-	-	-	-
151	, FileStack	9	1	0	0	69	0	0	-	-	0	-	-	-	-
152		1	1	0	0	69	0	0	-	-	0	-	-	-	-
153	, FileStackPtr	9	1	0	0	69	0	0	-	-	0	-	-	-	-
154		1	1	0	0	69	0	0	-	-	0	-	-	-	-
155	, MaxFiles	9	1	0	0	69	0	0	-	-	0	-	-	-	-
156		1	1	0	0	69	0	0	-	-	0	-	-	-	-
157	, MaxString	9	1	0	0	69	0	0	-	-	0	-	-	-	-
158		1	1	0	0	69	0	0	-	-	0	-	-	-	-
159	, Parse	9	1	0	0	69	0	0	-	-	0	-	-	-	-
160		1	1	0	0	69	0	0	-	-	0	-	-	-	-
161	, ScanInit	9	1	0	0	69	0	0	-	-	0	-	-	-	-
162		1	1	0	0	69	0	0	-	-	0	-	-	-	-
163] 139	139	1	0	0	64	0	0	-	-	0	-	-	-	-
164	, Process	9	1	0	0	64	0	0	-	-	0	-	-	-	-
165		1	1	0	0	64	64	0	-	-	0	-	-	-	-
166	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
167	[145	145	1	0	0	69	0	0	-	-	0	-	-	-	-
168	Detach	1	1	0	0	69	0	0	-	-	0	-	-	-	-
169		9	1	0	0	69	0	0	-	-	0	-	-	-	-
170	, priorityBackground	1	1	0	0	69	0	0	-	-	0	-	-	-	-
171		9	1	0	0	69	0	0	-	-	0	-	-	-	-
172	, SetPriority	1	1	0	0	69	0	0	-	-	0	-	-	-	-
173		9	1	0	0	69	0	0	-	-	0	-	-	-	-
174	, Yield	1	1	0	0	69	0	0	-	-	0	-	-	-	-
175] 139	139	1	0	0	64	0	0	-	-	0	-	-	-	-
176		9	1	0	0	64	0	0	-	-	0	-	-	-	-
177	, Put	1	1	0	0	64	64	0	-	-	0	-	-	-	-
178	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
179	[145	145	1	0	0	69	0	0	-	-	0	-	-	-	-
180	Text	1	1	0	0	69	0	0	-	-	0	-	-	-	-
181] 139	139	1	0	0	64	0	0	-	-	0	-	-	-	-
182		9	1	0	0	64	0	0	-	-	0	-	-	-	-
183	, Runtime	1	1	0	0	64	64	0	-	-	0	-	-	-	-
184	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
185	[145	145	1	0	0	69	0	0	-	-	0	-	-	-	-
186	BoundsFault	1	1	0	0	69	0	0	-	-	0	-	-	-	-
187		9	1	0	0	69	0	0	-	-	0	-	-	-	-
188	, GetTableBase	1	1	0	0	69	0	0	-	-	0	-	-	-	-
189] 139	139	1	0	0	64	0	0	-	-	0	-	-	-	-
190		9	1	0	0	64	0	0	-	-	0	-	-	-	-
191	, Stream	1	1	0	0	64	64	0	-	-	0	-	-	-	-
192	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
193	[145	145	1	0	0	69	0	0	-	-	0	-	-	-	-
194	Handle	1	1	0	0	69	0	0	-	-	0	-	-	-	-
195	, 9	9	1	0	0	69	0	0	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Smt Cnt	BEnd lv1	Proc lv1	List mark	id mark	Var mark	Var Scpe	Read Wrt	Decl Loca	Import Dp	Import.....Export Identifier	Type	Values
196	Position	1	1	0	0	69	0	0	-	-	0	-	-	-	-
197	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
198	SetPosition	1	1	0	0	69	0	0	-	-	0	-	-	-	-
199]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
200	,	9	1	0	0	64	0	0	-	-	0	-	-	-	-
201	String	1	1	0	0	64	64	0	-	-	0	-	-	-	-
202	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
203	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
204	AppendStringAndGrow	1	1	0	0	69	0	0	-	-	0	-	-	-	-
205	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
206	Copy	1	1	0	0	69	0	0	-	-	0	-	-	-	-
207	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
208	CopyToNewString	1	1	0	0	69	0	0	-	-	0	-	-	-	-
209	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
210	Empty	1	1	0	0	69	0	0	-	-	0	-	-	-	-
211	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
212	EquivalentSubStrings	1	1	0	0	69	0	0	-	-	0	-	-	-	-
213	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
214	FreeString	1	1	0	0	69	0	0	-	-	0	-	-	-	-
215	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
216	SubStringDescriptor	1	1	0	0	69	0	0	-	-	0	-	-	-	-
217]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
218	,	9	1	0	0	64	0	0	-	-	0	-	-	-	-
219	TextSW	1	1	0	0	64	64	0	-	-	0	-	-	-	-
220	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
221	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
222	ForceOutput	1	1	0	0	69	0	0	-	-	0	-	-	-	-
223]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
224	,	9	1	0	0	64	0	0	-	-	0	-	-	-	-
225	Token	1	1	0	0	64	64	0	-	-	0	-	-	-	-
226	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
227	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
228	FreeStringHandle	1	1	0	0	69	0	0	-	-	0	-	-	-	-
229	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
230	FreeTokenString	1	1	0	0	69	0	0	-	-	0	-	-	-	-
231	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
232	Handle	1	1	0	0	69	0	0	-	-	0	-	-	-	-
233	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
234	MaybeQuoted	1	1	0	0	69	0	0	-	-	0	-	-	-	-
235	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
236	StringToHandle	1	1	0	0	69	0	0	-	-	0	-	-	-	-
237	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
238	UnterminatedQuote	1	1	0	0	69	0	0	-	-	0	-	-	-	-
239]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
240	,	9	1	0	0	64	0	0	-	-	0	-	-	-	-
241	Tool	1	1	0	0	64	64	0	-	-	0	-	-	-	-
242	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
243	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
244	Create	1	1	0	0	69	0	0	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Write	Decl Loca	Import OpExport Identifier	Type	Values
245	, Destroy	9	1	0	0	69	0	0	-	-	0	-	-	-	-
246		1	1	0	0	69	0	0	-	-	0	-	-	-	-
247	, MakeFileSW	9	1	0	0	69	0	0	-	-	0	-	-	-	-
248		1	1	0	0	69	0	0	-	-	0	-	-	-	-
249	, MakeFormSW	9	1	0	0	69	0	0	-	-	0	-	-	-	-
250		1	1	0	0	69	0	0	-	-	0	-	-	-	-
251	, MakeSWsProc	9	1	0	0	69	0	0	-	-	0	-	-	-	-
252		1	1	0	0	69	0	0	-	-	0	-	-	-	-
253	, UnusedLogName	9	1	0	0	69	0	0	-	-	0	-	-	-	-
254		1	1	0	0	69	0	0	-	-	0	-	-	-	-
255]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
256		9	1	0	0	64	0	0	-	-	0	-	-	-	-
257	, ToolDriver	1	1	0	0	64	64	0	-	-	0	-	-	-	-
258	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
259	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
260	Address	1	1	0	0	69	0	0	-	-	0	-	-	-	-
261	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
262	, NoteSWs	1	1	0	0	69	0	0	-	-	0	-	-	-	-
263]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
264		9	1	0	0	64	0	0	-	-	0	-	-	-	-
265	, ToolWindow	1	1	0	0	64	64	0	-	-	0	-	-	-	-
266	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
267	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
268	Activate	1	1	0	0	69	0	0	-	-	0	-	-	-	-
269]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
270	,	9	1	0	0	64	0	0	-	-	0	-	-	-	-
271	, UserInput	1	1	0	0	64	64	0	-	-	0	-	-	-	-
272	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
273	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
274	UserAbort	1	1	0	0	69	0	0	-	-	0	-	-	-	-
275]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
276	,	9	1	0	0	64	0	0	-	-	0	-	-	-	-
277	, Utils	1	1	0	0	64	64	0	-	-	0	-	-	-	-
278	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
279	[145	1	0	0	69	0	0	-	-	0	-	-	-	-
280	CardToString	1	1	0	0	69	0	0	-	-	0	-	-	-	-
281	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
282	, CloseMetFile	1	1	0	0	69	0	0	-	-	0	-	-	-	-
283	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
284	, ClearMetFile	1	1	0	0	69	0	0	-	-	0	-	-	-	-
285		9	1	0	0	69	0	0	-	-	0	-	-	-	-
286	, MetFileLength	1	1	0	0	69	0	0	-	-	0	-	-	-	-
287	,	9	1	0	0	69	0	0	-	-	0	-	-	-	-
288	, OpenMetFile	1	1	0	0	69	0	0	-	-	0	-	-	-	-
289]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
290		9	1	0	0	64	0	0	-	-	0	-	-	-	-
291	, Window	1	1	0	0	64	64	0	-	-	0	-	-	-	-
292	USING	69	1	0	0	64	0	0	-	-	0	-	-	-	-
293	[145	1	0	0	69	0	0	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scope	Read Write	Decl Loca	Import Op	Export Identifier	Type	Values
294	Handle	1	1	0	0	69	0	0	-	-	0	-	-	-	-
295]	139	1	0	0	64	0	0	-	-	0	-	-	-	-
296	;	10	2	0	0	0	0	0	-	-	0	-	-	-	-
297	DesignExtractorImpl	1	2	0	0	0	44	0	-	-	0	-	-	-	-
298	;	11	2	0	0	0	0	0	-	-	0	-	-	-	-
299	MONITOR	44	2	0	0	0	0	0	-	-	0	-	-	-	-
300	IMPORTS	65	2	0	0	65	0	0	-	-	0	-	-	-	-
301	ChangeTable	1	2	0	0	65	65	0	-	-	0	-	-	-	-
302	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
303	Display	1	2	0	0	65	65	0	-	-	0	-	-	-	-
304	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
305	Exec	1	2	0	0	65	65	0	-	-	0	-	-	-	-
306	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
307	Format	1	2	0	0	65	65	0	-	-	0	-	-	-	-
308	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
309	FormSW	1	2	0	0	65	65	0	-	-	0	-	-	-	-
310	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
311	Heap	1	2	0	0	65	65	0	-	-	0	-	-	-	-
312	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
313	HeraldWindow	1	2	0	0	65	65	0	-	-	0	-	-	-	-
314	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
315	MFile	1	2	0	0	65	65	0	-	-	0	-	-	-	-
316	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
317	MesaTab	1	2	0	0	65	65	0	-	-	0	-	-	-	-
318	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
319	MSegment	1	2	0	0	65	65	0	-	-	0	-	-	-	-
320	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
321	MStream	1	2	0	0	65	65	0	-	-	0	-	-	-	-
322	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
323	Parser	1	2	0	0	65	65	0	-	-	0	-	-	-	-
324	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
325	Process	1	2	0	0	65	65	0	-	-	0	-	-	-	-
326	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
327	Put	1	2	0	0	65	65	0	-	-	0	-	-	-	-
328	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
329	Runtime	1	2	0	0	65	65	0	-	-	0	-	-	-	-
330	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
331	Stream	1	2	0	0	65	65	0	-	-	0	-	-	-	-
332	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
333	String	1	2	0	0	65	65	0	-	-	0	-	-	-	-
334	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
335	TextSW	1	2	0	0	65	65	0	-	-	0	-	-	-	-
336	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
337	Token	1	2	0	0	65	65	0	-	-	0	-	-	-	-
338	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
339	Tool	1	2	0	0	65	65	0	-	-	0	-	-	-	-
340	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-
341	ToolDriver	1	2	0	0	65	65	0	-	-	0	-	-	-	-
342	,	9	2	0	0	65	0	0	-	-	0	-	-	-	-

DESIGN EXTRATION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	8End lvl	Proc lvl	List mark	id mark	Var mark	Var Scape	Read Write	Decl Loca	Import Op	Import.....Export Identifier	Type	Values
343	ToolWindow	1	2	0	0	65	65	0	-	-	0	-	-	-	-
344	,	9	2	0	0	65	65	0	-	-	0	-	-	-	-
345	UserInput	1	2	0	0	65	65	0	-	-	0	-	-	-	-
346	,	9	2	0	0	65	65	0	-	-	0	-	-	-	-
347	Utils	1	2	0	0	65	65	0	-	-	0	-	-	-	-
348	EXPORTS	66	2	0	0	66	66	0	-	-	0	-	-	-	-
349	Utils	1	2	0	0	66	66	0	-	-	0	-	-	-	-
350	=	15	3	0	0	0	0	0	-	-	0	-	-	-	-
351	8EGIN	147	3	1	0	0	0	0	-	-	0	-	-	-	-
352	running	1	3	1	0	0	501	0	-	W	0	-	-	800LEAN	-
353	:	11	3	1	0	0	0	0	-	-	0	-	-	-	-
354	BOOLEAN	202	3	1	0	0	0	0	-	-	0	-	-	-	-
355		14	3	1	0	0	0	0	-	-	0	-	-	-	-
356	FALSE	205	3	1	0	0	0	0	-	-	0	-	-	-	-
357	;	10	4	1	0	0	0	0	-	-	0	-	-	-	-
358	created	1	4	1	0	0	501	0	-	W	0	-	-	800LEAN	-
359	:	11	4	1	0	0	0	0	-	-	0	-	-	-	-
360	BOOLEAN	202	4	1	0	0	0	0	-	-	0	-	-	-	-
361		14	4	1	0	0	0	0	-	-	0	-	-	-	-
362	FALSE	205	4	1	0	0	0	0	-	-	0	-	-	-	-
363	;	10	5	1	0	0	0	0	-	-	0	-	-	-	-
364	ExternalStackPtr	1	5	1	0	0	501	0	-	-	0	-	-	LDNG POINTER	LONG POINTER TO Exte
365	:	11	5	1	0	0	0	0	-	-	0	-	-	-	-
366	TYPE	49	5	1	0	0	0	0	-	-	0	-	-	-	-
367	=	15	5	1	0	0	0	0	-	-	0	-	-	-	-
368	LONG	0	5	1	0	0	0	0	-	-	0	-	-	-	-
369	POINTER	0	5	1	0	0	0	0	-	-	0	-	-	-	-
370	TO	0	5	1	0	0	0	0	-	-	0	-	-	-	-
371	ExternalStack	0	5	1	0	0	0	0	-	-	0	-	-	-	-
372	;	10	6	1	0	0	0	0	-	-	0	-	-	TYPE	ARRAY [0 .. maxExte
373	ExternalStack	1	6	1	0	0	501	0	-	-	0	-	-	-	-
374	:	11	6	1	0	0	0	0	-	-	0	-	-	-	-
375	TYPE	49	6	1	0	0	0	0	-	-	0	-	-	-	-
376	=	15	6	1	0	0	0	0	-	-	0	-	-	-	-
377	ARRAY	34	6	1	0	0	0	0	-	-	0	-	-	-	-
378	[145	6	1	0	201	0	0	-	-	0	-	-	-	-
379	0	2	6	1	0	201	0	0	-	-	0	-	-	-	-
380	..	12	6	1	0	201	0	0	-	-	0	-	-	-	-
381	maxExternal	1	6	1	0	201	500	0	-	-	0	-	-	-	-
382]	139	6	1	0	0	0	0	-	-	0	-	-	-	-
383	OF	55	6	1	0	0	0	0	-	-	0	-	-	-	-
384	ExternalProps	1	6	1	0	0	500	0	-	R	0	-	-	-	-
385	;	10	7	1	0	0	0	0	-	-	0	-	-	RECORD	extStoreNam
386	ExternalProps	1	7	1	0	0	501	0	-	-	0	-	-	-	-
387	:	11	7	1	0	0	0	0	-	-	0	-	-	-	-
388	TYPE	49	7	1	0	0	0	0	-	-	0	-	-	-	-
389	=	15	7	1	0	0	0	0	-	-	0	-	-	-	-
390	RECORD	0	7	1	0	0	0	0	-	-	0	-	-	-	-
391	[0	7	1	0	201	0	0	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS
=====

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Wrt	Decl Loca	Import Op	Identifiers	Export Identifier	Type	Values
392	extStoreName	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
393	;	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
394	LONG	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
395	STRING	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
396		0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
397	NIL	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
398		0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
399	ExtStoreLocation	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
400	;	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
401	CARDINAL	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
402		0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
403	0	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
404		0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
405	extStoreType	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
406	;	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
407	LONG	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
408	STRING	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
409		0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
410	NIL	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
411		0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
412	extStoreAction	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
413	;	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
414	LONG	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
415	STRING	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
416		0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
417	NIL	0	7	1	0	201	0	0	-	-	0	-	-	-	-	-
418]	139	7	1	0	0	0	0	-	-	0	-	-	-	-	-
419	;	10	8	1	0	0	0	0	-	-	0	-	-	-	-	-
420	maxExternal	1	8	1	0	0	501	0	-	W	0	-	-	-	CARDINAL	-
421	;	11	8	1	0	0	0	0	-	-	0	-	-	-	-	-
422	CARDINAL	203	8	1	0	0	0	0	-	-	0	-	-	-	-	-
423	=	15	9	1	0	0	0	0	-	-	0	-	-	-	-	-
424	30	2	9	1	0	0	0	0	-	-	0	-	-	-	-	-
425	;	10	10	1	0	0	0	0	-	-	0	-	-	-	-	-
426	eSCount	1	10	1	0	0	501	0	-	W	0	-	-	-	CARDINAL	-
427	;	11	10	1	0	0	0	0	-	-	0	-	-	-	-	-
428	CARDINAL	203	10	1	0	0	0	0	-	-	0	-	-	-	-	-
429		14	10	1	0	0	0	0	-	-	0	-	-	-	-	-
430	0	2	10	1	0	0	0	0	-	-	0	-	-	-	-	-
431	;	10	11	1	0	0	0	0	-	-	0	-	-	-	-	-
432	fileStackPtr	1	11	1	0	0	501	0	-	-	0	-	I Parser	-	FileStackPtr	-
433	;	11	11	1	0	0	0	0	-	-	0	-	-	-	-	-
434	Parser	0	11	1	0	0	0	0	-	-	0	-	-	-	-	-
435	;	27	11	1	0	0	0	0	-	-	0	-	-	-	-	-
436	FileStackPtr	0	11	1	0	0	0	0	-	-	0	-	-	-	-	-
437	;	10	12	1	0	0	0	0	-	-	0	-	-	-	-	-
438	fileCount	1	12	1	0	0	501	0	-	W	0	-	-	-	CARDINAL	-
439	;	11	12	1	0	0	0	0	-	-	0	-	-	-	-	-
440	CARDINAL	203	12	1	0	0	0	0	-	-	0	-	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scape	Read Wrt	Decl Loca	Import OpExport Identifier	Type	Values
441	0	14	12	1	0	0	0	0	-	-	0	-	-	-	-
442	;	2	12	1	0	0	0	0	-	-	0	-	-	-	-
443	extStackPtr	10	13	1	0	0	0	0	-	-	0	-	-	ExternalStackPt	-
444	;	1	13	1	0	0	501	0	-	-	0	-	-	-	-
445	;	11	13	1	0	0	0	0	-	-	0	-	-	-	-
446	ExternalStackPtr	0	13	1	0	0	0	0	-	-	0	-	-	-	-
447	;	10	14	1	0	0	0	0	-	-	0	-	-	-	-
448	atomStackPtr	1	14	1	0	0	501	0	-	-	0	I	Parser	AtomStackPtr	-
449	;	11	14	1	0	0	0	0	-	-	0	-	-	-	-
450	Parser	0	14	1	0	0	0	0	-	-	0	-	-	-	-
451	;	27	14	1	0	0	0	0	-	-	0	-	-	-	-
452	AtomStackPtr	0	14	1	0	0	0	0	-	-	0	-	-	-	-
453	;	10	15	1	0	0	0	0	-	-	0	-	-	-	-
454	atomIndexPtr	1	15	1	0	0	501	0	-	-	0	I	Parser	AtomIndexPtr	-
455	;	11	15	1	0	0	0	0	-	-	0	-	-	-	-
456	Parser	0	15	1	0	0	0	0	-	-	0	-	-	-	-
457	;	27	15	1	0	0	0	0	-	-	0	-	-	-	-
458	AtomIndexPtr	0	15	1	0	0	0	0	-	-	0	-	-	-	-
459	;	10	16	1	0	0	0	0	-	-	0	-	-	-	-
460	MaxFiles	1	16	1	0	0	501	0	-	-	0	-	-	CARDINAL	-
461	;	11	16	1	0	0	0	0	-	-	0	-	-	-	-
462	CARDINAL	203	16	1	0	0	0	0	-	-	0	-	-	-	-
463	=	15	17	1	0	0	0	0	-	-	0	-	-	-	-
464	30	2	17	1	0	0	0	0	-	-	0	-	-	-	-
465	;	10	18	1	0	0	0	0	-	-	0	-	-	-	-
466	sh	1	18	1	0	0	501	0	-	-	0	I	MSegment	Handle	-
467	;	11	18	1	0	0	0	0	-	-	0	-	-	-	-
468	MSegment	0	18	1	0	0	0	0	-	-	0	-	-	-	-
469	;	27	18	1	0	0	0	0	-	-	0	-	-	-	-
470	Handle	0	18	1	0	0	0	0	-	-	0	-	-	-	-
471	;	10	19	1	0	0	0	0	-	-	0	-	-	-	-
472	zone	1	19	1	0	0	501	0	P	W	0	-	-	UNCOUNTED_ZONE	-
473	;	11	19	1	0	0	0	0	-	-	0	-	-	-	-
474	PUBLIC	70	19	1	0	0	0	0	-	-	0	-	-	-	-
475	UNCOUNTED_ZONE	53	19	1	0	0	0	0	-	-	0	-	-	-	-
476	;	46	19	1	0	0	0	0	-	-	0	-	-	-	-
477	Heap	14	19	1	0	0	0	0	-	-	0	-	-	-	-
478	;	0	19	1	0	0	0	0	-	R	0	-	-	-	-
479	systemZone	27	19	1	0	0	0	0	-	R	0	-	Heap	-	-
480	;	1	19	1	0	0	500	0	M	R	478	-	-	-	-
481	;	10	20	1	0	0	0	0	-	-	0	-	-	-	-
482	Enter	1	20	1	1	0	37	1	-	-	0	-	-	-	-
483	;	11	20	1	1	0	0	0	-	-	0	-	-	-	-
484	ENTRY	72	20	1	1	0	0	0	-	-	0	-	-	-	-
485	PROC	38	20	1	1	0	0	0	-	-	0	-	-	-	-
486	RETURNS	57	20	1	1	0	0	0	-	-	0	-	-	-	-
487	[145	20	1	1	57	0	0	-	-	0	-	-	-	-
488	ok	1	20	1	1	57	500	1	-	-	0	-	-	BOOL	-
489	;	11	20	1	1	57	0	1	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS
=====

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Wrt	Decl Loca	Import OpExport Identifier	Type	Values
490	800L	0	20	1	1	57	0	1	-	-	0	-	-	-	-
491]	139	20	1	1	0	0	1	-	-	0	-	-	-	-
492	=	15	21	1	1	0	0	1	-	-	0	-	-	-	-
493	BEGIN	147	21	2	1	0	0	1	-	-	0	-	-	-	-
494	ENABLE	135	21	2	1	0	0	1	-	-	0	-	-	-	-
495	UNWIND	0	21	2	1	0	0	1	-	-	0	-	-	-	-
496	=>	13	21	2	1	0	0	1	-	-	0	-	-	-	-
497	NULL	103	21	2	1	0	0	1	-	-	0	-	-	-	-
498	;	10	22	2	1	0	0	1	-	-	0	-	-	-	-
499	RETURN	117	22	2	1	0	0	1	-	-	0	-	-	-	-
500	[145	22	2	1	117	0	1	-	-	0	-	-	-	-
501	ok	1	22	2	1	117	500	1	R	-	488	-	-	-	-
502	:	11	22	2	1	117	0	1	-	-	0	-	-	-	-
503	IF	104	22	2	1	117	0	1	-	-	0	-	-	-	-
504	running	1	22	2	1	117	500	1	G	-	352	-	-	-	-
505	THEN	105	22	2	1	117	0	1	-	-	0	-	-	-	-
506	FALSE	205	22	2	1	117	0	1	-	-	0	-	-	-	-
507	ELSE	106	22	2	1	117	0	1	-	-	0	-	-	-	-
508	(144	22	2	1	117	0	1	-	-	0	-	-	-	-
509	running	1	22	2	1	117	500	1	G	W	352	-	-	-	-
510		14	22	2	1	117	0	1	-	-	0	-	-	-	-
511	TRUE	204	22	2	1	117	0	1	-	-	0	-	-	-	-
512)	138	22	2	1	117	0	1	-	-	0	-	-	-	-
513]	139	22	2	1	0	0	1	-	-	0	-	-	-	-
514	:	10	23	2	1	0	0	1	-	-	0	-	-	-	-
515	END	141	23	1	0	0	0	0	-	-	0	-	-	-	-
516	;	10	24	1	0	0	0	0	-	-	0	-	-	-	-
517	Exit	1	24	1	1	0	37	2	-	-	0	-	-	-	-
518	;	11	24	1	1	0	0	2	-	-	0	-	-	-	-
519	ENTRY	72	24	1	1	0	0	2	-	-	0	-	-	-	-
520	PROC	38	24	1	1	0	0	2	-	-	0	-	-	-	-
521	=	15	25	1	1	0	0	2	-	-	0	-	-	-	-
522	BEGIN	147	25	2	1	0	0	2	-	-	0	-	-	-	-
523	ENABLE	135	25	2	1	0	0	2	-	-	0	-	-	-	-
524	UNWIND	0	25	2	1	0	0	2	-	-	0	-	-	-	-
525	=>	13	25	2	1	0	0	2	-	-	0	-	-	-	-
526	NULL	103	25	2	1	0	0	2	-	-	0	-	-	-	-
527	;	10	26	2	1	0	0	2	-	-	0	-	-	-	-
528	running	1	26	2	1	0	500	2	G	W	352	-	-	-	-
529		14	26	2	1	0	0	2	-	-	0	-	-	-	-
530	FALSE	205	26	2	1	0	0	2	-	-	0	-	-	-	-
531	;	10	27	2	1	0	0	2	-	-	0	-	-	-	-
532	END	141	27	1	0	0	0	0	-	-	0	-	-	-	-
533	;	10	28	1	0	0	0	0	-	-	0	-	-	-	-
534	permZone	1	28	1	0	0	501	0	-	-	0	-	-	UNCOUNTED ZONE	NIL
535	,	9	28	1	0	0	0	0	-	-	0	-	-	-	-
536	tempZone	1	28	1	0	0	501	0	P	W	0	-	-	UNCOUNTED ZONE	-
537	;	11	28	1	0	0	0	0	-	-	0	-	-	-	-
538	PUBLIC	70	28	1	0	0	0	0	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	8End lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Write	Decl Loca	Import Op	Import.....Export Identifier	Type	Values
539	UNCOUNTED	53	28	1	0	0	0	0	-	-	0	-	-	-	-
540	ZONE	46	28	1	0	0	0	0	-	-	0	-	-	-	-
541		14	28	1	0	0	0	0	-	-	0	-	-	-	-
542	NIL	0	28	1	0	0	0	0	-	-	0	-	-	-	-
543	;	10	29	1	0	0	0	0	-	-	0	-	-	-	-
544	window	1	29	1	0	0	501	0	P	-	0	I	Window	Handle	NIL
545	,	9	29	1	0	0	0	0	-	-	0	-	-	-	-
546	formSW	1	29	1	0	0	501	0	P	-	0	I	Window	Handle	NIL
547	,	9	29	1	0	0	0	0	-	-	0	-	-	-	-
548	fileSW	1	29	1	0	0	501	0	P	-	0	I	Window	Handle	NIL
549	;	11	29	1	0	0	0	0	-	-	0	-	-	-	-
550	PUBLIC	70	29	1	0	0	0	0	-	-	0	-	-	-	-
551	Window	0	29	1	0	0	0	0	-	-	0	-	-	-	-
552	.	27	29	1	0	0	0	0	-	-	0	-	-	-	-
553	Handle	0	29	1	0	0	0	0	-	-	0	-	-	-	-
554		14	29	1	0	0	0	0	-	-	0	-	-	-	-
555	NIL	0	29	1	0	0	0	0	-	-	0	-	-	-	-
556	;	10	30	1	0	0	0	0	-	-	0	-	-	-	-
557	SCGraph	1	30	1	0	0	501	0	P	W	0	-	-	800LEAN	-
558	;	11	30	1	0	0	0	0	-	-	0	-	-	-	-
559	PUBLIC	70	30	1	0	0	0	0	-	-	0	-	-	-	-
560	800LEAN	202	30	1	0	0	0	0	-	-	0	-	-	-	-
561		14	30	1	0	0	0	0	-	-	0	-	-	-	-
562	TRUE	204	30	1	0	0	0	0	-	-	0	-	-	-	-
563	;	10	31	1	0	0	0	0	-	-	0	-	-	-	-
564	STable	1	31	1	0	0	501	0	P	W	0	-	-	800LEAN	-
565	;	11	31	1	0	0	0	0	-	-	0	-	-	-	-
566	PUBLIC	70	31	1	0	0	0	0	-	-	0	-	-	-	-
567	800LEAN	202	31	1	0	0	0	0	-	-	0	-	-	-	-
568		14	31	1	0	0	0	0	-	-	0	-	-	-	-
569	FALSE	205	31	1	0	0	0	0	-	-	0	-	-	-	-
570	;	10	32	1	0	0	0	0	-	-	0	-	-	-	-
571	DFDVariable	1	32	1	0	0	501	0	P	W	0	-	-	800LEAN	-
572	;	11	32	1	0	0	0	0	-	-	0	-	-	-	-
573	PUBLIC	70	32	1	0	0	0	0	-	-	0	-	-	-	-
574	800LEAN	202	32	1	0	0	0	0	-	-	0	-	-	-	-
575		14	32	1	0	0	0	0	-	-	0	-	-	-	-
576	FALSE	205	32	1	0	0	0	0	-	-	0	-	-	-	-
577	;	10	33	1	0	0	0	0	-	-	0	-	-	-	-
578	DFDTable	1	33	1	0	0	501	0	P	W	0	-	-	800LEAN	-
579	;	11	33	1	0	0	0	0	-	-	0	-	-	-	-
580	PUBLIC	70	33	1	0	0	0	0	-	-	0	-	-	-	-
581	800LEAN	202	33	1	0	0	0	0	-	-	0	-	-	-	-
582		14	33	1	0	0	0	0	-	-	0	-	-	-	-
583	TRUE	204	33	1	0	0	0	0	-	-	0	-	-	-	-
584	;	10	34	1	0	0	0	0	-	-	0	-	-	-	-
585	rawData	1	34	1	0	0	501	0	P	W	0	-	-	800LEAN	-
586	;	11	34	1	0	0	0	0	-	-	0	-	-	-	-
587	PUBLIC	70	34	1	0	0	0	0	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Wrte	Decl Loca	Import OpExport Identifier	Type	Values
588	BOOLEAN	202	34	1	0	0	0	0	-	-	-	0	-	-	-
589		14	34	1	0	0	0	0	-	-	-	0	-	-	-
590	FALSE	205	34	1	0	0	0	0	-	-	-	0	-	-	-
591	;	10	35	1	0	0	0	0	-	-	-	0	-	-	-
592	spare	1	35	1	0	0	501	0	P	W	0	0	-	BOOLEAN	-
593	;	11	35	1	0	0	0	0	-	-	-	0	-	-	-
594	PUBLIC	70	35	1	0	0	0	0	-	-	-	0	-	-	-
595	BOOLEAN	202	35	1	0	0	0	0	-	-	-	0	-	-	-
596		14	35	1	0	0	0	0	-	-	-	0	-	-	-
597	FALSE	205	35	1	0	0	0	0	-	-	-	0	-	-	-
598	;	10	36	1	0	0	0	0	-	-	-	0	-	-	-
599	debugg	1	36	1	0	0	501	0	P	W	0	0	-	BOOLEAN	-
600	;	11	36	1	0	0	0	0	-	-	-	0	-	-	-
601	PUBLIC	70	36	1	0	0	0	0	-	-	-	0	-	-	-
602	BOOLEAN	202	36	1	0	0	0	0	-	-	-	0	-	-	-
603		14	36	1	0	0	0	0	-	-	-	0	-	-	-
604	FALSE	205	36	1	0	0	0	0	-	-	-	0	-	-	-
605	;	10	37	1	0	0	0	0	-	-	-	0	-	-	-
606	repeats	1	37	1	0	0	501	0	P	W	0	0	-	BOOLEAN	-
607	;	11	37	1	0	0	0	0	-	-	-	0	-	-	-
608	PUBLIC	70	37	1	0	0	0	0	-	-	-	0	-	-	-
609	BOOLEAN	202	37	1	0	0	0	0	-	-	-	0	-	-	-
610		14	37	1	0	0	0	0	-	-	-	0	-	-	-
611	FALSE	205	37	1	0	0	0	0	-	-	-	0	-	-	-
612	;	10	38	1	0	0	0	0	-	-	-	0	-	-	-
613	backend	1	38	1	0	0	501	0	F	W	0	0	-	BOOLEAN	-
614	;	11	38	1	0	0	0	0	-	-	-	0	-	-	-
615	PUBLIC	70	38	1	0	0	0	0	-	-	-	0	-	-	-
616	BOOLEAN	202	38	1	0	0	0	0	-	-	-	0	-	-	-
617		14	38	1	0	0	0	0	-	-	-	0	-	-	-
618	FALSE	205	38	1	0	0	0	0	-	-	-	0	-	-	-
619	;	10	39	1	0	0	0	0	-	-	-	0	-	-	-
620	encode	1	39	1	0	0	501	0	F	W	0	0	-	BOOLEAN	-
621	;	11	39	1	0	0	0	0	-	-	-	0	-	-	-
622	PUBLIC	70	39	1	0	0	0	0	-	-	-	0	-	-	-
623	BOOLEAN	202	39	1	0	0	0	0	-	-	-	0	-	-	-
624		14	39	1	0	0	0	0	-	-	-	0	-	-	-
625	FALSE	205	39	1	0	0	0	0	-	-	-	0	-	-	-
626	;	10	40	1	0	0	0	0	-	-	-	0	-	-	-
627	background	1	40	1	0	0	501	0	F	W	0	0	-	BOOLEAN	-
628	;	11	40	1	0	0	0	0	-	-	-	0	-	-	-
629	PUBLIC	70	40	1	0	0	0	0	-	-	-	0	-	-	-
630	BOOLEAN	202	40	1	0	0	0	0	-	-	-	0	-	-	-
631		14	40	1	0	0	0	0	-	-	-	0	-	-	-
632	TRUE	204	40	1	0	0	0	0	-	-	-	0	-	-	-
633	;	10	41	1	0	0	0	0	-	-	-	0	-	-	-
634	overlap	1	41	1	0	0	501	0	P	W	0	0	-	CARDINAL	-
635	;	11	41	1	0	0	0	0	-	-	-	0	-	-	-
636	PUBLIC	70	41	1	0	0	0	0	-	-	-	0	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scape	Read Wrt	Decl Loca	Import Op	Identifiers	Export	Type	Values
637	CARDINAL	203	41	1	0	0	0	0	-	-	0	-	-	-	-	-
638	10	14	41	1	0	0	0	0	-	-	0	-	-	-	-	-
639	;	2	41	1	0	0	0	0	-	-	0	-	-	-	-	-
640	totalErrors	10	42	1	0	0	0	0	-	-	0	-	-	-	CARDINAL	0
641	totalWarnings	1	42	1	0	0	501	0	-	-	0	-	-	-	CARDINAL	-
642	;	9	42	1	0	0	0	0	-	-	0	-	-	-	-	-
643	totalWarnings	1	42	1	0	0	501	0	-	W	0	-	-	-	CARDINAL	-
644	;	11	42	1	0	0	0	0	-	-	0	-	-	-	-	-
645	LONG	48	42	1	0	0	0	0	-	-	0	-	-	-	-	-
646	CARDINAL	203	42	1	0	0	0	0	-	-	0	-	-	-	-	-
647	10	14	42	1	0	0	0	0	-	-	0	-	-	-	-	-
648	;	0	42	1	0	0	0	0	-	-	0	-	-	-	-	-
649	;	10	43	1	0	0	0	0	-	-	0	-	-	-	-	-
650	toolName	1	43	1	0	0	501	0	-	W	0	-	-	-	LONG STRING	-
651	;	11	43	1	0	0	0	0	-	-	0	-	-	-	-	-
652	LONG	48	43	1	0	0	0	0	-	-	0	-	-	-	-	-
653	STRING	208	43	1	0	0	0	0	-	-	0	-	-	-	-	-
654	NIL	14	43	1	0	0	0	0	-	-	0	-	-	-	-	-
655	;	101	43	1	0	0	0	0	-	-	0	-	-	-	-	-
656	tablePtr	10	44	1	0	0	0	0	-	-	0	-	-	-	-	-
657	;	1	44	1	0	0	501	0	-	-	0	-	-	-	TableRef	NIL
658	ParseTable	11	44	1	0	0	0	0	-	-	0	-	-	-	-	-
659	;	0	44	1	0	0	0	0	-	-	0	-	-	-	-	-
660	TableRef	27	44	1	0	0	0	0	-	-	0	-	-	-	-	-
661	;	0	44	1	0	0	0	0	-	-	0	-	-	-	-	-
662	NIL	14	44	1	0	0	0	0	-	-	0	-	-	-	-	-
663	;	0	44	1	0	0	0	0	-	-	0	-	-	-	-	-
664	execCommand	10	45	1	0	0	0	0	-	-	0	-	-	-	LONG STRING	-
665	;	1	45	1	0	0	501	0	-	W	0	-	-	-	-	-
666	LONG	11	45	1	0	0	0	0	-	-	0	-	-	-	-	-
667	STRING	48	45	1	0	0	0	0	-	-	0	-	-	-	-	-
668	=	208	45	1	0	0	0	0	-	-	0	-	-	-	-	-
669	"DesignExtractor	15	46	1	0	0	0	0	-	-	0	-	-	-	-	-
670	;	6	46	1	0	0	0	0	-	-	0	-	-	-	-	-
671	filename	10	47	1	0	0	0	0	-	-	0	-	-	-	LONG STRING	-
672	;	1	47	1	0	0	501	0	-	W	0	-	-	-	-	-
673	LONG	11	47	1	0	0	0	0	-	-	0	-	-	-	-	-
674	STRING	48	47	1	0	0	0	0	-	-	0	-	-	-	-	-
675	NIL	208	47	1	0	0	0	0	-	-	0	-	-	-	-	-
676	;	14	47	1	0	0	0	0	-	-	0	-	-	-	-	-
677	ErrorCode	101	47	1	0	0	0	0	-	-	0	-	-	-	-	-
678	;	10	48	1	0	0	0	0	-	-	0	-	-	-	TYPE	-
679	TYPE	1	48	1	0	0	501	0	-	-	0	-	-	-	-	-
680	=	11	48	1	0	0	0	0	-	-	0	-	-	-	-	-
681	{	49	48	1	0	0	0	0	-	-	0	-	-	-	-	-
682	Other	15	48	1	0	0	0	0	-	-	0	-	-	-	-	-
683	;	146	48	2	0	0	0	0	-	-	0	-	-	-	-	-
684	;	1	48	2	0	0	500	0	-	R	0	-	-	-	-	-
685	}	140	48	1	0	0	0	0	-	-	0	-	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scope	Read Wrt	Decl Loca	Import Op	Export Identifier	Type	Values
686	;	10	49	1	0	0	0	0	-	-	0	-	-	LONG STRING	-
687	PublicError	1	49	1	0	0	501	0	P	W	0	-	-	-	-
688	;	11	49	1	0	0	0	0	-	-	0	-	-	-	-
689	PUBLIC	70	49	1	0	0	0	0	-	-	0	-	-	-	-
690	SIGNAL	40	49	1	0	0	0	0	-	-	0	-	-	-	-
691	[145	49	1	0	201	0	0	-	-	0	-	-	-	-
692	error	1	49	1	0	201	501	0	-	-	0	-	-	ErrorCode	NIL
693	:	11	49	1	0	201	0	0	-	-	0	-	-	-	-
694	ErrorCode	0	49	1	0	201	0	0	-	-	0	-	-	-	-
695	,	9	49	1	0	201	0	0	-	-	0	-	-	-	-
696	string	1	49	1	0	201	201	0	-	-	0	-	-	LONG STRING	-
697	;	11	49	1	0	201	0	0	-	-	0	-	-	-	-
698	LONG	48	49	1	0	201	0	0	-	-	0	-	-	-	-
699	STRING	208	49	1	0	201	0	0	-	-	0	-	-	-	-
700	-	14	49	1	0	201	0	0	-	-	0	-	-	-	-
701	NIL	0	49	1	0	201	0	0	-	-	0	-	-	-	-
702]	139	49	1	0	0	0	0	-	-	0	-	-	-	-
703	=	15	50	1	0	0	0	0	-	-	0	-	-	-	-
704	CODE	76	50	1	0	0	0	0	-	-	0	-	-	-	-
705	;	10	51	1	0	0	0	0	-	-	0	-	-	-	-
706	log	1	51	1	1	0	37	3	P	-	0	P	Format	StringProc	-
707	:	11	51	1	1	0	0	0	-	-	0	-	-	-	-
708	PUBLIC	70	51	1	1	0	0	3	-	-	0	-	-	-	-
709	Format	0	51	1	1	0	0	0	-	-	0	-	-	-	-
710	.	27	51	1	1	0	0	3	-	-	0	-	-	-	-
711	StringProc	0	51	1	1	0	0	0	-	-	0	-	-	-	-
712	=	15	52	1	1	0	0	3	-	-	0	-	-	-	-
713	BEGIN	147	52	2	1	0	0	3	-	-	0	-	-	-	-
714	Put	0	52	2	1	0	0	3	-	R	0	-	-	-	-
715	.	27	52	2	1	0	0	3	-	-	0	-	-	-	-
716	Text	1	52	2	1	0	237	3	M	-	714	-	Put	-	-
717	[145	52	2	1	238	0	3	-	-	0	-	-	-	-
718	fileSW	1	52	2	1	238	500	3	P	-	548	-	-	-	-
719	,	9	52	2	1	238	0	3	-	-	0	-	-	-	-
720	s	1	52	2	1	238	500	3	M	-	0	-	-	-	-
721]	139	52	2	1	0	0	3	-	-	0	-	-	-	-
722	;	10	53	2	1	0	0	3	-	-	0	-	-	-	-
723	END	141	53	1	0	0	0	0	-	-	0	-	-	-	-
724	;	10	54	1	0	0	0	0	-	-	0	-	-	-	-
725	ExecutiveProc	1	54	1	1	0	37	4	-	-	0	I	Exec	ExecProc	-
726	:	11	54	1	1	0	0	4	-	-	0	-	-	-	-
727	Exec	0	54	1	1	0	0	4	-	-	0	-	-	-	-
728	.	27	54	1	1	0	0	4	-	-	0	-	-	-	-
729	ExecProc	0	54	1	1	0	0	4	-	-	0	-	-	-	-
730	=	15	55	1	1	0	0	4	-	-	0	-	-	-	-
731	BEGIN	147	55	2	1	0	0	4	-	-	0	-	-	-	-
732	ToolWindow	0	55	2	1	0	0	4	-	R	0	-	-	-	-
733	.	27	55	2	1	0	0	4	-	-	0	-	-	-	-
734	Activate	1	55	2	1	0	237	4	M	-	732	-	ToolWindow	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Write	Decl Loca	Import Op	Identifiers	Export Identifier	Type	Values
735	[145	55	2	1	238	0	4	-	-	0	-	-	-	-	-
736	window	1	55	2	1	238	500	4	P	-	544	-	-	-	-	-
737]	139	55	2	1	0	0	4	-	-	0	-	-	-	-	-
738	;	10	56	2	1	0	0	4	-	-	0	-	-	-	-	-
739	END	141	56	1	0	0	0	0	-	-	0	-	-	-	-	-
740	;	10	57	1	0	0	0	0	-	-	0	-	-	-	-	-
741	Unload	1	57	1	1	0	37	5	-	-	0	I	Exec	-	ExecProc	-
742	;	11	57	1	1	0	0	5	-	-	0	-	-	-	-	-
743	Exec	0	57	1	1	0	0	5	-	-	0	-	-	-	-	-
744	.	27	57	1	1	0	0	5	-	-	0	-	-	-	-	-
745	ExecProc	0	57	1	1	0	0	5	-	-	0	-	-	-	-	-
746	=	15	58	1	1	0	0	5	-	-	0	-	-	-	-	-
747	BEGIN	147	58	2	1	0	0	5	-	-	0	-	-	-	-	-
748	IF	104	58	2	1	0	0	5	-	-	0	-	-	-	-	-
749	-	21	58	2	1	0	0	5	-	-	0	-	-	-	-	-
750	Enter	1	58	2	1	0	237	5	-	-	482	-	-	-	-	-
751	[145	58	2	1	238	0	5	-	-	0	-	-	-	-	-
752]	139	58	2	1	0	0	5	-	-	0	-	-	-	-	-
753	THEN	105	58	2	1	0	0	5	-	-	0	-	-	-	-	-
754	BEGIN	147	58	3	1	0	0	5	-	-	0	-	-	-	-	-
755	Format	0	58	3	1	0	0	5	-	R	0	-	-	-	-	-
756	.	27	58	3	1	0	0	5	-	-	0	-	-	-	-	-
757	Text	1	58	3	1	0	237	5	M	-	755	-	Format	-	-	-
758	[145	58	3	1	238	0	5	-	-	0	-	-	-	-	-
759	Exec	0	58	3	1	238	238	5	-	-	0	-	-	-	-	-
760	.	27	58	3	1	238	0	5	-	-	0	-	-	-	-	-
761	OutputProc	1	58	3	1	238	237	5	M	-	759	-	Exec	-	-	-
762	[145	58	3	1	238	0	5	-	-	0	-	-	-	-	-
763	h	1	58	3	1	238	500	5	M	-	0	-	-	-	-	-
764]	139	58	3	1	0	0	5	-	-	0	-	-	-	-	-
765	.	9	58	3	1	0	0	5	-	-	0	-	-	-	-	-
766	"DesignExtractor is	6	58	3	1	0	0	5	-	-	0	-	-	-	-	-
767]	139	58	3	1	0	0	5	-	-	0	-	-	-	-	-
768	;	10	59	3	1	0	0	5	-	-	0	-	-	-	-	-
769	RETURN	117	59	3	1	0	0	5	-	-	0	-	-	-	-	-
770	[145	59	3	1	117	0	5	-	-	0	-	-	-	-	-
771	abort	1	59	3	1	117	500	5	M	-	0	-	-	-	-	-
772]	139	59	3	1	0	0	5	-	-	0	-	-	-	-	-
773	;	10	60	3	1	0	0	5	-	-	0	-	-	-	-	-
774	END	141	60	2	1	0	0	5	-	-	0	-	-	-	-	-
775	ELSE	106	60	2	1	0	0	5	-	-	0	-	-	-	-	-
776	BEGIN	147	60	3	1	0	0	5	-	-	0	-	-	-	-	-
777	ENABLE	135	60	3	1	0	0	5	-	-	0	-	-	-	-	-
778	UNWIND	0	60	3	1	0	0	5	-	-	0	-	-	-	-	-
779	=>	13	60	3	1	0	0	5	-	-	0	-	-	-	-	-
780	Exit	1	60	3	1	0	237	5	-	-	517	-	-	-	-	-
781	[145	60	3	1	238	0	5	-	-	0	-	-	-	-	-
782]	139	60	3	1	0	0	5	-	-	0	-	-	-	-	-
783	;	10	61	3	1	0	0	5	-	-	0	-	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scope	Read Write	Decl Loca	Import Op	Export Identifier	Type	Values
784	IF	104	61	3	1	0	0	5	-	-	0	-	-	-	-
785	(144	61	3	1	0	0	5	-	-	0	-	-	-	-
786	window	1	61	3	1	0	500	5	P	R	544	-	-	-	-
787	#	16	61	3	1	0	0	5	-	-	0	-	-	-	-
788	NIL	101	61	3	1	0	0	5	-	-	0	-	-	-	-
789)	138	61	3	1	0	0	5	-	-	0	-	-	-	-
790	THEN	105	61	3	1	0	0	5	-	-	0	-	-	-	-
791	BEGIN	147	61	4	1	0	0	5	-	-	0	-	-	-	-
792	Tool	0	61	4	1	0	0	5	-	R	0	-	-	-	-
793	.	27	61	4	1	0	0	5	-	-	0	-	-	-	-
794	Destroy	1	61	4	1	0	237	5	M	-	792	-	Tool	-	-
795	[145	61	4	1	238	0	5	-	-	0	-	-	-	-
796	window	1	61	4	1	238	500	5	P	-	544	-	-	-	-
797]	139	61	4	1	0	0	5	-	-	0	-	-	-	-
798	;	10	62	4	1	0	0	5	-	-	0	-	-	-	-
799	window	1	62	4	1	0	500	5	P	W	544	-	-	-	-
800		14	62	4	1	0	0	5	-	-	0	-	-	-	-
801	NIL	101	62	4	1	0	0	5	-	-	0	-	-	-	-
802	;	10	63	4	1	0	0	5	-	-	0	-	-	-	-
803	END	141	63	3	1	0	0	5	-	-	0	-	-	-	-
804	;	10	64	3	1	0	0	5	-	-	0	-	-	-	-
805	Exec	0	64	3	1	0	0	5	-	R	0	-	-	-	-
806	.	27	64	3	1	0	0	5	-	-	0	-	-	-	-
807	RemoveCommand	1	64	3	1	0	237	5	M	-	805	-	Exec	-	-
808	[145	64	3	1	238	0	5	-	-	0	-	-	-	-
809	h	1	64	3	1	238	500	5	M	-	0	-	-	-	-
810	;	9	64	3	1	238	0	5	-	-	0	-	-	-	-
811	execCommand	1	64	3	1	238	500	5	G	-	665	-	-	-	-
812]	139	64	3	1	0	0	5	-	-	0	-	-	-	-
813	;	10	65	3	1	0	0	5	-	-	0	-	-	-	-
814	IF	104	65	3	1	0	0	5	-	-	0	-	-	-	-
815	(144	65	3	1	0	0	5	-	-	0	-	-	-	-
816	tempZone	1	65	3	1	0	500	5	P	R	536	-	-	-	-
817	#	16	65	3	1	0	0	5	-	-	0	-	-	-	-
818	NIL	101	65	3	1	0	0	5	-	-	0	-	-	-	-
819)	138	65	3	1	0	0	5	-	-	0	-	-	-	-
820	THEN	105	65	3	1	0	0	5	-	-	0	-	-	-	-
821	BEGIN	147	65	4	1	0	0	5	-	-	0	-	-	-	-
822	Heap	0	65	4	1	0	0	5	-	R	0	-	-	-	-
823	.	27	65	4	1	0	0	5	-	-	0	-	-	-	-
824	Delete	1	65	4	1	0	237	5	M	-	822	-	Heap	-	-
825	[145	65	4	1	238	0	5	-	-	0	-	-	-	-
826	tempZone	1	65	4	1	238	500	5	P	-	536	-	-	-	-
827]	139	65	4	1	0	0	5	-	-	0	-	-	-	-
828	;	10	66	4	1	0	0	5	-	-	0	-	-	-	-
829	tempZone	1	66	4	1	0	500	5	P	W	536	-	-	-	-
830		14	66	4	1	0	0	5	-	-	0	-	-	-	-
831	NIL	101	66	4	1	0	0	5	-	-	0	-	-	-	-
832	;	10	67	4	1	0	0	5	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Write	Decl Loca	Import,.....Export Identifier	Type	Values
833	END	141	67	3	1	0	0	5	-	-	0	-	-	-
834	;	10	68	3	1	0	0	5	-	-	0	-	-	-
835	IF	104	68	3	1	0	0	5	-	-	0	-	-	-
836	(144	68	3	1	0	0	5	-	-	0	-	-	-
837	permZone	1	68	3	1	0	500	5	G	R	534	-	-	-
838	#	16	68	3	1	0	0	5	-	-	0	-	-	-
839	NIL	101	68	3	1	0	0	5	-	-	0	-	-	-
840)	138	68	3	1	0	0	5	-	-	0	-	-	-
841	THEN	105	68	3	1	0	0	5	-	-	0	-	-	-
842	BEGIN	147	68	4	1	0	0	5	-	-	0	-	-	-
843	Heap	0	68	4	1	0	0	5	-	R	0	-	-	-
844	.	27	68	4	1	0	0	5	-	-	0	-	-	-
845	Delete	1	68	4	1	0	237	5	M	-	843	- Heap	-	-
846	[145	68	4	1	238	0	5	-	-	0	-	-	-
847	permZone	1	68	4	1	238	500	5	G	-	534	-	-	-
848]	139	68	4	1	0	0	5	-	-	0	-	-	-
849	;	10	69	4	1	0	0	5	-	-	0	-	-	-
850	permZone	1	69	4	1	0	500	5	G	W	534	-	-	-
851	NIL	14	69	4	1	0	0	5	-	-	0	-	-	-
852	;	101	69	4	1	0	0	5	-	-	0	-	-	-
853	;	10	70	4	1	0	0	5	-	-	0	-	-	-
854	END	141	70	3	1	0	0	5	-	-	0	-	-	-
855	;	10	71	3	1	0	0	5	-	-	0	-	-	-
856	IF	104	71	3	1	0	0	5	-	-	0	-	-	-
857	(144	71	3	1	0	0	5	-	-	0	-	-	-
858	atomStackPtr	1	71	3	1	0	500	5	G	R	448	-	-	-
859	#	16	71	3	1	0	0	5	-	-	0	-	-	-
860	NIL	101	71	3	1	0	0	5	-	-	0	-	-	-
861)	138	71	3	1	0	0	5	-	-	0	-	-	-
862	THEN	105	71	3	1	0	0	5	-	-	0	-	-	-
863	BEGIN	147	71	4	1	0	0	5	-	-	0	-	-	-
864	FreeAtomStack	1	71	4	1	0	237	5	-	-	3618	-	-	-
865	[145	71	4	1	238	0	5	-	-	0	-	-	-
866	atomStackPtr	1	71	4	1	238	500	5	G	-	3622	-	-	-
867	,	9	71	4	1	238	0	5	-	-	0	-	-	-
868	sh	1	71	4	1	238	500	5	G	-	3628	-	-	-
869]	139	71	4	1	0	0	5	-	-	0	-	-	-
870	;	10	72	4	1	0	0	5	-	-	0	-	-	-
871	END	141	72	3	1	0	0	5	-	-	0	-	-	-
872	;	10	73	3	1	0	0	5	-	-	0	-	-	-
873	IF	104	73	3	1	0	0	5	-	-	0	-	-	-
874	(144	73	3	1	0	0	5	-	-	0	-	-	-
875	fileStackPtr	1	73	3	1	0	500	5	G	R	432	-	-	-
876	#	16	73	3	1	0	0	5	-	-	0	-	-	-
877	NIL	101	73	3	1	0	0	5	-	-	0	-	-	-
878)	138	73	3	1	0	0	5	-	-	0	-	-	-
879	THEN	105	73	3	1	0	0	5	-	-	0	-	-	-
880	BEGIN	147	73	4	1	0	0	5	-	-	0	-	-	-
881	zone	1	73	4	1	0	0	5	-	R	0	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Wrt	Decl Loca	Import Op	Export Identifier	Type	Values
882	.	27	73	4	1	0	0	5	-	-	0	-	-	-	-
883	FREE	0	73	4	1	0	0	5	-	-	0	-	-	-	-
884	[145	73	4	1	201	0	5	-	-	0	-	-	-	-
885	@	28	73	4	1	201	0	5	-	-	0	-	-	-	-
886	fileStackPtr	1	73	4	1	201	500	5	G	-	432	-	-	-	-
887]	139	73	4	1	0	0	5	-	-	0	-	-	-	-
888	;	10	74	4	1	0	0	5	-	-	0	-	-	-	-
889	END	141	74	3	1	0	0	5	-	-	0	-	-	-	-
890	;	10	75	3	1	0	0	5	-	-	0	-	-	-	-
891	IF	104	75	3	1	0	0	5	-	-	0	-	-	-	-
892	(144	75	3	1	0	0	5	-	-	0	-	-	-	-
893	atomIndexPtr	1	75	3	1	0	500	5	G	R	454	-	-	-	-
894	#	16	75	3	1	0	0	5	-	-	0	-	-	-	-
895	NIL	101	75	3	1	0	0	5	-	-	0	-	-	-	-
896)	138	75	3	1	0	0	5	-	-	0	-	-	-	-
897	THEN	105	75	3	1	0	0	5	-	-	0	-	-	-	-
898	BEGIN	147	75	4	1	0	0	5	-	-	0	-	-	-	-
899	zone	1	75	4	1	0	0	5	-	R	0	-	-	-	-
900	.	27	75	4	1	0	0	5	-	-	0	-	-	-	-
901	FREE	0	75	4	1	0	0	5	-	-	0	-	-	-	-
902	[145	75	4	1	201	0	5	-	-	0	-	-	-	-
903	@	28	75	4	1	201	0	5	-	-	0	-	-	-	-
904	atomIndexPtr	1	75	4	1	201	500	5	G	-	454	-	-	-	-
905]	139	75	4	1	0	0	5	-	-	0	-	-	-	-
906	;	10	76	4	1	0	0	5	-	-	0	-	-	-	-
907	END	141	76	3	1	0	0	5	-	-	0	-	-	-	-
908	;	10	77	3	1	0	0	5	-	-	0	-	-	-	-
909	END	141	77	2	1	0	0	5	-	-	0	-	-	-	-
910	;	10	78	2	1	0	0	5	-	-	0	-	-	-	-
911	Exit	1	78	2	1	0	237	5	-	-	517	-	-	-	-
912	[145	78	2	1	238	0	5	-	-	0	-	-	-	-
913]	139	78	2	1	0	0	5	-	-	0	-	-	-	-
914	;	10	79	2	1	0	0	5	-	-	0	-	-	-	-
915	END	141	79	1	0	0	0	0	-	-	0	-	-	-	-
916	;	10	80	1	0	0	0	0	-	-	0	-	-	-	-
917	CheckForAbort	1	80	1	1	0	37	6	P	-	0	-	-	-	-
918	;	11	80	1	1	0	0	6	-	-	0	-	-	-	-
919	PUBLIC	70	80	1	1	0	0	6	-	-	0	-	-	-	-
920	PROC	38	80	1	1	0	0	6	-	-	0	-	-	-	-
921	=	15	81	1	1	0	0	6	-	-	0	-	-	-	-
922	BEGIN	147	81	2	1	0	0	6	-	-	0	-	-	-	-
923	IF	104	81	2	1	0	0	6	-	-	0	-	-	-	-
924	UserInput	0	81	2	1	0	0	6	-	R	0	-	-	-	-
925	.	27	81	2	1	0	0	6	-	-	0	-	-	-	-
926	UserAbort	1	81	2	1	0	237	6	M	-	924	-	UserInput	-	-
927	[145	81	2	1	238	0	6	-	-	0	-	-	-	-
928	window	1	81	2	1	238	500	6	P	-	544	-	-	-	-
929]	139	81	2	1	0	0	6	-	-	0	-	-	-	-
930	THEN	105	81	2	1	0	0	6	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	8End lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Write	Decl Loca	Import OpExport Identifier	Type	Values
931	ERROR	41	81	2	1	0	0	6	-	-	0	-	-	-	-
932	ABORTED	0	81	2	1	0	0	6	-	-	0	-	-	-	-
933	;	10	82	2	1	0	0	6	-	-	0	-	-	-	-
934	END	141	82	1	0	0	0	0	-	-	0	-	-	-	-
935	;	10	83	1	0	0	0	0	-	-	0	-	-	-	-
936	ProcessFile	1	83	1	1	0	37	7	-	-	0	I	MFile	EnumerateProc	-
937	;	11	83	1	1	0	0	7	-	-	0	-	-	-	-
938	MFile	0	83	1	1	0	0	7	-	-	0	-	-	-	-
939	;	27	83	1	1	0	0	7	-	-	0	-	-	-	-
940	EnumerateProc	0	83	1	1	0	0	7	-	-	0	-	-	-	-
941	=	15	84	1	1	0	0	7	-	-	0	-	-	-	-
942	8EGIN	147	84	2	1	0	0	7	-	-	0	-	-	-	-
943	file	1	84	2	1	0	501	7	E	-	0	I	MStream	Handle	NIL
944	;	11	84	2	1	0	0	7	-	-	0	-	-	-	-
945	MStream	0	84	2	1	0	0	7	-	-	0	-	-	-	-
946	;	27	84	2	1	0	0	7	-	-	0	-	-	-	-
947	Handle	0	84	2	1	0	0	7	-	-	0	-	-	-	-
948	NIL	14	84	2	1	0	0	7	-	-	0	-	-	-	-
949	;	0	84	2	1	0	0	7	-	-	0	-	-	-	-
950	;	10	85	2	1	0	0	7	-	-	0	-	-	-	-
951	errors	1	85	2	1	0	501	7	-	-	0	-	-	CARDINAL	0
952	;	9	85	2	1	0	0	7	-	-	0	-	-	-	-
953	warnings	1	85	2	1	0	501	7	-	-	0	-	-	CARDINAL	-
954	;	11	85	2	1	0	0	7	-	-	0	-	-	-	-
955	CARDINAL	203	85	2	1	0	0	7	-	-	0	-	-	-	-
956	;	14	85	2	1	0	0	7	-	-	0	-	-	-	-
957	0	0	85	2	1	0	0	7	-	-	0	-	-	-	-
958	;	10	86	2	1	0	0	7	-	-	0	-	-	-	-
959	parseZone	1	86	2	1	0	501	7	-	-	0	-	-	UNCOUNTED ZONE	-
960	;	11	86	2	1	0	0	7	-	-	0	-	-	-	-
961	UNCOUNTED	53	86	2	1	0	0	7	-	-	0	-	-	-	-
962	ZONE	46	86	2	1	0	0	7	-	-	0	-	-	-	-
963	;	14	86	2	1	0	0	7	-	-	0	-	-	-	-
964	NIL	101	86	2	1	0	0	7	-	-	0	-	-	-	-
965	;	10	87	2	1	0	0	7	-	-	0	-	-	-	-
966	tempString	1	87	2	1	0	501	7	-	-	0	-	-	LONG STRING	-
967	;	11	87	2	1	0	0	7	-	-	0	-	-	-	-
968	LONG	48	87	2	1	0	0	7	-	-	0	-	-	-	-
969	STRING	208	87	2	1	0	0	7	-	-	0	-	-	-	-
970	;	10	88	2	1	0	0	7	-	-	0	-	-	-	-
971	CleanUp	1	88	2	2	0	37	8	-	-	0	-	-	-	-
972	;	11	88	2	2	0	0	8	-	-	0	-	-	-	-
973	PROCEDURE	37	88	2	2	0	0	8	-	-	0	-	-	-	-
974	[145	88	2	2	200	500	8	-	-	0	-	-	800LEAN	-
975	ok	1	88	2	2	200	0	8	-	-	0	-	-	-	-
976	;	11	88	2	2	200	0	8	-	-	0	-	-	-	-
977	BOOLEAN	202	88	2	2	200	0	8	-	-	0	-	-	-	-
978	;	14	88	2	2	200	0	8	-	-	0	-	-	-	-
979	TRUE	204	88	2	2	200	0	8	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS

=====

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Wrte	Decl Loca	Import Op	Identifrier	Export	Type	Values
980]	139	88	2	2	0	0	8	-	-	0	-	-	-	-	-
981	=	15	89	2	2	0	0	8	-	-	0	-	-	-	-	-
982	BEGIN	147	89	3	2	0	0	8	-	-	0	-	-	-	-	-
983	tempASP	1	89	3	2	0	501	8	-	W	0	-	-	-	CARDINAL	-
984	:	11	89	3	2	0	0	8	-	-	0	-	-	-	-	-
985	CARDINAL	203	89	3	2	0	0	8	-	-	0	-	-	-	-	-
986	-	14	89	3	2	0	0	8	-	-	0	-	-	-	-	-
987	0	2	89	3	2	0	0	8	-	-	0	-	-	-	-	-
988	;	10	90	3	2	0	0	8	-	-	0	-	-	-	-	-
989	IF	104	90	3	2	0	0	8	-	-	0	-	-	-	-	-
990	file	1	90	3	2	0	500	8	E	R	943	-	-	-	-	-
991	#	16	90	3	2	0	0	8	-	-	0	-	-	-	-	-
992	NIL	101	90	3	2	0	0	8	-	-	0	-	-	-	-	-
993	THEN	105	90	3	2	0	0	8	-	-	0	-	-	-	-	-
994	{	146	90	4	2	0	0	8	-	-	0	-	-	-	-	-
995	IF	104	90	4	2	0	0	8	-	-	0	-	-	-	-	-
996	file	0	90	4	2	0	0	8	-	R	0	-	-	-	-	-
997	.	27	90	4	2	0	0	8	-	-	0	-	-	-	-	-
998	delete	1	90	4	2	0	500	8	M	R	996	-	file	-	-	-
999	#	16	90	4	2	0	0	8	-	-	0	-	-	-	-	-
1000	NIL	101	90	4	2	0	0	8	-	-	0	-	-	-	-	-
1001	THEN	105	90	4	2	0	0	8	-	-	0	-	-	-	-	-
1002	file	0	90	4	2	0	0	8	-	R	0	-	-	-	-	-
1003	.	27	90	4	2	0	0	8	-	-	0	-	-	-	-	-
1004	delete	1	90	4	2	0	237	8	M	-	1002	-	file	-	-	-
1005	[145	90	4	2	238	0	8	-	-	0	-	-	-	-	-
1006	file	1	90	4	2	238	500	8	E	-	943	-	-	-	-	-
1007]	139	90	4	2	0	0	8	-	-	0	-	-	-	-	-
1008	:	10	91	4	2	0	0	8	-	-	0	-	-	-	-	-
1009	file	1	91	4	2	0	500	8	E	W	943	-	-	-	-	-
1010	-	14	91	4	2	0	0	8	-	-	0	-	-	-	-	-
1011	NIL	101	91	4	2	0	0	8	-	-	0	-	-	-	-	-
1012	}	140	91	3	2	0	0	8	-	-	0	-	-	-	-	-
1013	;	10	92	3	2	0	0	8	-	-	0	-	-	-	-	-
1014	log	1	92	3	2	0	237	8	-	-	706	-	-	-	-	-
1015	[145	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1016	SELECT	149	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1017	TRUE	204	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1018	FROM	108	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1019	-	21	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1020	ok	1	92	3	2	238	500	8	I	-	975	-	-	-	-	-
1021	=>	13	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1022	.	6	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1023	,	9	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1024	errors	1	92	3	2	238	500	8	L	-	951	-	-	-	-	-
1025	>	18	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1026	0	2	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1027	=>	13	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1028	.	6	92	3	2	238	0	8	-	-	0	-	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scape	Read Wrte	Decl Loca	Import Op	Identfier	Export	Type	Values
1029	;	9	92	3	2	238	0	8	-	-	-	0	-	-	-	-
1030	warnings	1	92	3	2	238	500	8	L	-	953	-	-	-	-	-
1031	>	18	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1032	0	2	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1033	=>	13	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1034	"	6	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1035	;	9	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1036	ENDCASE	143	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1037	=>	13	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1038	;	6	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1039]	139	92	3	2	238	0	8	-	-	0	-	-	-	-	-
1040	;	10	93	3	2	0	0	8	-	-	0	-	-	-	-	-
1041	log	1041	1	93	3	2	0	237	8	-	706	-	-	-	-	-
1042	[145	93	3	2	238	0	8	-	-	0	-	-	-	-	-
1043	" Cumulative Bu	6	93	3	2	238	0	8	-	-	0	-	-	-	-	-
1044]	139	93	3	2	0	0	8	-	-	0	-	-	-	-	-
1045	;	10	94	3	2	0	0	8	-	-	0	-	-	-	-	-
1046	log	1	94	3	2	0	500	8	M	R	0	-	-	-	-	-
1047	[145	94	3	2	201	0	8	-	-	0	-	-	-	-	-
1048	Utils	0	94	3	2	201	201	8	-	-	0	-	-	-	-	-
1049	;	27	94	3	2	201	0	8	-	-	0	-	-	-	-	-
1050	CardToString	1	94	3	2	201	500	8	M	-	1048	-	Utils	-	-	-
1051	[145	94	3	2	201	0	8	-	-	0	-	-	-	-	-
1052	fileStackPtr	0	94	3	2	201	500	8	G	-	432	-	-	-	-	-
1053	[145	94	3	2	201	0	8	-	-	0	-	-	-	-	-
1054	fileCount	1	94	3	2	201	500	8	G	-	438	-	-	-	-	-
1055]	139	94	3	2	0	0	8	-	-	0	-	-	-	-	-
1056	;	27	94	3	2	0	0	8	-	-	0	-	-	-	-	-
1057	endIndex	1	94	3	2	0	500	8	D	R	1052	-	fileStackPtr	-	-	-
1058]	139	94	3	2	0	0	8	-	-	0	-	-	-	-	-
1059]	139	94	3	2	0	0	8	-	-	0	-	-	-	-	-
1060	;	10	95	3	2	0	0	8	-	-	0	-	-	-	-	-
1061	log	1	95	3	2	0	237	8	-	-	706	-	-	-	-	-
1062	[145	95	3	2	238	0	8	-	-	0	-	-	-	-	-
1063	"	6	95	3	2	238	0	8	-	-	0	-	-	-	-	-
1064]	139	95	3	2	0	0	8	-	-	0	-	-	-	-	-
1065	;	10	96	3	2	0	0	8	-	-	0	-	-	-	-	-
1066	log	1	96	3	2	0	237	8	-	-	706	-	-	-	-	-
1067	[145	96	3	2	238	0	8	-	-	0	-	-	-	-	-
1068	Utils	0	96	3	2	238	238	8	-	-	0	-	-	-	-	-
1069	;	27	96	3	2	238	0	8	-	-	0	-	-	-	-	-
1070	CardToString	1	96	3	2	238	237	8	M	-	1068	-	Utils	-	-	-
1071	[145	96	3	2	238	0	8	-	-	0	-	-	-	-	-
1072	fileCount	1	96	3	2	238	500	8	G	-	438	-	-	-	-	-
1073]	139	96	3	2	0	0	8	-	-	0	-	-	-	-	-
1074]	139	96	3	2	0	0	8	-	-	0	-	-	-	-	-
1075	;	10	97	3	2	0	0	8	-	-	0	-	-	-	-	-
1076	log	1	97	3	2	0	237	8	-	-	706	-	-	-	-	-
1077	[145	97	3	2	238	0	8	-	-	0	-	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	8End lvl	Proc lvl	List mark	id mark	Var mark	Var Scope	Read Wrt	Decl Loca	Import Op	Export Identifier	Type	Values
1078		6	97	3	2	238	0	8	-	-	0	-	-	-	-
1079]	139	97	3	2	0	0	8	-	-	0	-	-	-	-
1080	;	10	98	3	2	0	0	8	-	-	0	-	-	-	-
1081	log	1	98	3	2	0	237	8	-	-	706	-	-	-	-
1082	[145	98	3	2	238	0	8	-	-	0	-	-	-	-
1083	" LOC this file	6	98	3	2	238	0	8	-	-	0	-	-	-	-
1084]	139	98	3	2	0	0	8	-	-	0	-	-	-	-
1085	;	10	99	3	2	0	0	8	-	-	0	-	-	-	-
1086	log	1	99	3	2	0	500	8	M	R	0	-	-	-	-
1087	[145	99	3	2	201	0	8	-	-	0	-	-	-	-
1088	Utils	0	99	3	2	201	201	8	-	-	0	-	-	-	-
1089	.	27	99	3	2	201	0	8	-	-	0	-	-	-	-
1090	CardToCString	1	99	3	2	201	500	8	M	-	1088	-	Utils	-	-
1091	[145	99	3	2	201	0	8	-	-	0	-	-	-	-
1092	atomStackPtr	0	99	3	2	201	500	8	G	-	448	-	-	-	-
1093	[145	99	3	2	201	0	8	-	-	0	-	-	-	-
1094	atomIndexPtr	1	99	3	2	201	500	8	G	-	454	-	-	-	-
1095	.	26	99	3	2	201	0	8	-	-	0	-	-	-	-
1096]	139	99	3	2	0	0	8	-	-	0	-	-	-	-
1097	.	27	99	3	2	0	0	8	-	-	0	-	-	-	-
1098	sCount	1	99	3	2	0	500	8	D	R	1092	-	atomStackPtr	-	-
1099]	139	99	3	2	0	0	8	-	-	0	-	-	-	-
1100]	139	99	3	2	0	0	8	-	-	0	-	-	-	-
1101	;	10	100	3	2	0	0	8	-	-	0	-	-	-	-
1102	log	1	100	3	2	0	237	8	-	-	706	-	-	-	-
1103	[145	100	3	2	238	0	8	-	-	0	-	-	-	-
1104	"	6	100	3	2	238	0	8	-	-	0	-	-	-	-
1105]	139	100	3	2	0	0	8	-	-	0	-	-	-	-
1106	;	10	101	3	2	0	0	8	-	-	0	-	-	-	-
1107	totalErrors	1	101	3	2	0	500	8	G	W	641	-	-	-	-
1108	-	14	101	3	2	0	0	8	-	-	0	-	-	-	-
1109	totalErrors	1	101	3	2	0	500	8	G	R	641	-	-	-	-
1110	+	22	101	3	2	0	0	8	-	-	0	-	-	-	-
1111	errors	1	101	3	2	0	500	8	L	R	951	-	-	-	-
1112	;	10	102	3	2	0	0	8	-	-	0	-	-	-	-
1113	totalWarnings	1	102	3	2	0	500	8	G	W	643	-	-	-	-
1114	-	14	102	3	2	0	0	8	-	-	0	-	-	-	-
1115	totalWarnings	1	102	3	2	0	500	8	G	R	643	-	-	-	-
1116	+	22	102	3	2	0	0	8	-	-	0	-	-	-	-
1117	warnings	1	102	3	2	0	500	8	L	R	953	-	-	-	-
1118	;	10	103	3	2	0	0	8	-	-	0	-	-	-	-
1119	END	141	103	2	1	0	0	7	-	-	0	-	-	-	-
1120	;	10	104	2	1	0	0	7	-	-	0	-	-	-	-
1121	8EGIN	147	104	3	1	0	0	7	-	-	0	-	-	-	-
1122	ENABLE	135	104	3	1	0	0	7	-	-	0	-	-	-	-
1123	8EGIN	147	104	4	1	0	0	7	-	-	0	-	-	-	-
1124	MFile	0	104	4	1	0	0	7	-	R	0	-	-	-	-
1125	.	27	104	4	1	0	0	7	-	-	0	-	-	-	-
1126	Error	1	104	4	1	0	500	7	M	R	1124	-	MFile	-	-

DESIGN EXTRACTION STATISTICS
=====

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scope	Read Write	Decl Loca	Import Op	Export Identifier	Type	Values
1127	, MStream	9	104	4	1	0	0	7	-	-	0	-	-	-	-
1128	MStream	0	104	4	1	0	0	7	-	R	0	-	-	-	-
1129	, Error	27	104	4	1	0	0	7	-	-	0	-	-	-	-
1130	Error	1	104	4	1	0	500	7	M	R	1128	-	MStream	-	-
1131	, Runtime	9	104	4	1	0	0	7	-	-	0	-	-	-	-
1132	Runtime	0	104	4	1	0	0	7	-	R	0	-	-	-	-
1133	, BoundsFault	27	104	4	1	0	0	7	-	-	0	-	-	-	-
1134	BoundsFault	1	104	4	1	0	500	7	M	R	1132	-	Runtime	-	-
1135	=>	13	104	4	1	0	0	7	-	-	0	-	-	-	-
1136	BEGIN	147	104	5	1	0	0	7	-	-	0	-	-	-	-
1137	log	1	104	5	1	0	237	7	-	-	706	-	-	-	-
1138	["Unknown file pr	145	104	5	1	238	0	7	-	-	0	-	-	-	-
1139]	6	104	5	1	238	0	7	-	-	0	-	-	-	-
1140		139	104	5	1	0	0	7	-	-	0	-	-	-	-
1141	;	10	105	5	1	0	0	7	-	-	0	-	-	-	-
1142	GOTO	120	105	5	1	0	0	7	-	-	0	-	-	-	-
1143	error	1	105	5	1	0	500	7	G	R	692	-	-	-	-
1144	END	141	105	4	1	0	0	7	-	-	0	-	-	-	-
1145	;	10	106	4	1	0	0	7	-	-	0	-	-	-	-
1146	UNWIND	0	106	4	1	0	0	7	-	-	0	-	-	-	-
1147	=>	13	106	4	1	0	0	7	-	-	0	-	-	-	-
1148	CleanUp	1	106	4	1	0	237	7	-	-	971	-	-	-	-
1149	[145	106	4	1	238	0	7	-	-	0	-	-	-	-
1150	ok	1	106	4	1	238	238	7	-	-	0	-	-	-	-
1151	;	11	106	4	1	238	0	7	-	-	0	-	-	-	-
1152	FALSE	205	106	4	1	238	0	7	-	-	0	-	-	-	-
1153]	139	106	4	1	0	0	7	-	-	0	-	-	-	-
1154	;	10	107	4	1	0	0	7	-	-	0	-	-	-	-
1155	END	141	107	3	1	0	0	7	-	-	0	-	-	-	-
1156	;	10	108	3	1	0	0	7	-	-	0	-	-	-	-
1157	CheckForAbort	1	108	3	1	0	237	7	-	-	917	-	-	-	-
1158	[145	108	3	1	238	0	7	-	-	0	-	-	-	-
1159]	139	108	3	1	0	0	7	-	-	0	-	-	-	-
1160	;	10	109	3	1	0	0	7	-	-	0	-	-	-	-
1161	file	1	109	3	1	0	500	7	E	W	943	-	-	-	-
1162	MStream	14	109	3	1	0	0	7	-	R	0	-	-	-	-
1163		0	109	3	1	0	0	7	-	-	0	-	-	-	-
1164	, Create	27	109	3	1	0	0	7	-	-	0	-	MStream	-	-
1165	Create	1	109	3	1	0	237	7	M	-	1163	-	-	-	-
1166	[145	109	3	1	238	0	7	-	-	0	-	-	-	-
1167	file	1	109	3	1	238	501	7	-	-	0	-	-	fileProc	-
1168	;	11	109	3	1	238	0	7	-	-	0	-	-	-	-
1169	fileProc	0	109	3	1	238	0	7	-	-	0	-	-	-	-
1170	[145	109	3	1	238	0	7	-	-	0	-	-	-	-
1171	access	1	109	3	1	238	501	7	-	-	0	-	-	readOnly	-
1172	;	11	109	3	1	238	0	7	-	-	0	-	-	-	-
1173	readOnly	0	109	3	1	238	0	7	-	-	0	-	-	-	-
1174	,	9	109	3	1	238	0	7	-	-	0	-	-	-	-
1175	release	1	109	3	1	238	238	7	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Wrte	Decl Loca	Import OpExport Identifier	Type	Values
1176	:	11	109	3	1	238	0	7	-	-	0	-	-	-	-
1177	[145	109	3	1	238	0	7	-	-	0	-	-	-	-
1178]	139	109	3	1	0	0	7	-	-	0	-	-	-	-
1179]	139	109	3	1	0	0	7	-	-	0	-	-	-	-
1180	,	9	109	3	1	0	0	7	-	-	0	-	-	-	-
1181	release	1	109	3	1	0	500	7	M	R	0	-	-	-	-
1182	:	11	109	3	1	0	0	7	-	-	0	-	-	-	-
1183	[145	109	3	1	201	0	7	-	-	0	-	-	-	-
1184]	139	109	3	1	0	0	7	-	-	0	-	-	-	-
1185	,	29	109	3	1	0	0	7	-	-	0	-	-	-	-
1186	MFile	0	109	3	1	0	0	7	-	R	0	-	-	-	-
1187	:	27	109	3	1	0	0	7	-	-	0	-	-	-	-
1188	Error	1	109	3	1	0	500	7	M	R	1186	-	MFile	-	-
1189	,	9	109	3	1	0	0	7	-	-	0	-	-	-	-
1190	MStream	0	109	3	1	0	0	7	-	R	0	-	-	-	-
1191	:	27	109	3	1	0	0	7	-	-	0	-	-	-	-
1192	Error	1	109	3	1	0	500	7	M	R	1190	-	MStream	-	-
1193	=>	13	109	3	1	0	0	7	-	-	0	-	-	-	-
1194	{	146	109	4	1	0	0	7	-	-	0	-	-	-	-
1195	log	1	109	4	1	0	237	7	-	-	706	-	-	-	-
1196	[145	109	4	1	238	0	7	-	-	0	-	-	-	-
1197	"Cannot open inp	6	109	4	1	238	0	7	-	-	0	-	-	-	-
1198]	139	109	4	1	0	0	7	-	-	0	-	-	-	-
1199	:	10	110	4	1	0	0	7	-	-	0	-	-	-	-
1200	Format	0	110	4	1	0	0	7	-	R	0	-	-	-	-
1201	,	27	110	4	1	0	0	7	-	-	0	-	-	-	-
1202	Line	1	110	4	1	0	237	7	M	-	1200	-	Format	-	-
1203	[145	110	4	1	238	0	7	-	-	0	-	-	-	-
1204	log	1	110	4	1	238	500	7	M	-	0	-	-	-	-
1205	,	9	110	4	1	238	0	7	-	-	0	-	-	-	-
1206	name	1	110	4	1	238	500	7	M	-	0	-	-	-	-
1207]	139	110	4	1	0	0	7	-	-	0	-	-	-	-
1208	:	10	111	4	1	0	0	7	-	-	0	-	-	-	-
1209	GOTO	120	111	4	1	0	0	7	-	-	0	-	-	-	-
1210	error	1	111	4	1	0	500	7	G	R	692	-	-	-	-
1211	}	140	111	3	1	0	0	7	-	-	0	-	-	-	-
1212]	139	111	3	1	0	0	7	-	-	0	-	-	-	-
1213	:	10	112	3	1	0	0	7	-	-	0	-	-	-	-
1214	log	1	112	3	1	0	237	7	-	-	706	-	-	-	-
1215	[145	112	3	1	238	0	7	-	-	0	-	-	-	-
1216	"	6	112	3	1	238	0	7	-	-	0	-	-	-	-
1217]	139	112	3	1	0	0	7	-	-	0	-	-	-	-
1218	:	10	113	3	1	0	0	7	-	-	0	-	-	-	-
1219	log	1	113	3	1	0	237	7	-	-	706	-	-	-	-
1220	[145	113	3	1	238	0	7	-	-	0	-	-	-	-
1221	name	1	113	3	1	238	500	7	M	-	0	-	-	-	-
1222]	139	113	3	1	0	0	7	-	-	0	-	-	-	-
1223	:	10	114	3	1	0	0	7	-	-	0	-	-	-	-
1224	log	1	114	3	1	0	237	7	-	-	706	-	-	-	-

DESIGN EXTRACTION STATISTICS
=====

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scape	Read Wrte	Decl Loca	Import,.....Export Identifier	Type	Values
1225	[145	114	3	1	238	0	7	-	-	0	-	-	-
1226	"	6	114	3	1	238	0	7	-	-	0	-	-	-
1227]	139	114	3	1	0	0	7	-	-	0	-	-	-
1228	;	10	115	3	1	0	0	7	-	-	0	-	-	-
1229	fileCount	1	115	3	1	0	500	7	G	W	438	-	-	-
1230	fileCount	14	115	3	1	0	0	7	-	-	0	-	-	-
1231	fileCount	1	115	3	1	0	500	7	G	R	438	-	-	-
1232	+	22	115	3	1	0	0	7	-	-	0	-	-	-
1233	1	2	115	3	1	0	0	7	-	-	0	-	-	-
1234	;	10	116	3	1	0	0	7	-	-	0	-	-	-
1235	IF	104	116	3	1	0	0	7	-	-	0	-	-	-
1236	fileCount	1	116	3	1	0	500	7	G	R	438	-	-	-
1237	>	18	116	3	1	0	0	7	-	-	0	-	-	-
1238	Parser	0	116	3	1	0	0	7	-	R	0	-	-	-
1239	.	27	116	3	1	0	0	7	-	-	0	-	-	-
1240	MaxFiles	1	116	3	1	0	500	7	G	R	460	-	Parser	-
1241	THEN	105	116	3	1	0	0	7	-	-	0	-	-	-
1242	{	146	116	4	1	0	0	7	-	-	0	-	-	-
1243	log	1	116	4	1	0	237	7	-	-	706	-	-	-
1244	[145	116	4	1	238	0	7	-	-	0	-	-	-
1245	nnWarning - DesignE	6	116	4	1	238	0	7	-	-	0	-	-	-
1246]	139	116	4	1	0	0	7	-	-	0	-	-	-
1247	;	10	117	4	1	0	0	7	-	-	0	-	-	-
1248	tempString	1	117	4	1	0	500	7	L	W	966	-	-	-
1249	.	14	117	4	1	0	0	7	-	R	0	-	-	-
1250	String	0	117	4	1	0	0	7	-	-	0	-	-	-
1251	.	27	117	4	1	0	0	7	-	-	0	-	-	-
1252	CopyToNewString	1	117	4	1	0	237	7	M	-	1250	-	String	-
1253	[145	117	4	1	238	0	7	-	-	0	-	-	-
1254	Utils	0	117	4	1	238	238	7	-	-	0	-	-	-
1255	.	27	117	4	1	238	0	7	-	-	0	-	-	-
1256	CardToString	1	117	4	1	238	237	7	M	-	1254	-	Utils	-
1257	[145	117	4	1	238	0	7	-	-	0	-	-	-
1258	Parser	0	117	4	1	238	238	7	-	-	0	-	-	-
1259	.	27	117	4	1	238	0	7	-	-	0	-	-	-
1260	MaxFiles	1	117	4	1	238	500	7	G	-	460	-	Parser	-
1261]	139	117	4	1	0	0	7	-	-	0	-	-	-
1262	,	9	117	4	1	0	0	7	-	-	0	-	-	-
1263	zone	1	117	4	1	0	500	7	P	R	472	-	-	-
1264]	139	117	4	1	0	0	7	-	-	0	-	-	-
1265	;	10	118	4	1	0	0	7	-	-	0	-	-	-
1266	log	1	118	4	1	0	237	7	-	-	706	-	-	-
1267	[145	118	4	1	238	0	7	-	-	0	-	-	-
1268	tempString	1	118	4	1	238	500	7	L	-	966	-	-	-
1269]	139	118	4	1	0	0	7	-	-	0	-	-	-
1270	;	10	119	4	1	0	0	7	-	-	0	-	-	-
1271	String	0	119	4	1	0	0	7	-	R	0	-	-	-
1272	.	27	119	4	1	0	0	7	-	-	0	-	-	-
1273	FreeString	1	119	4	1	0	237	7	M	-	1271	-	String	-

DESIGN EXTRACTION STATISTICS

Ind #	Atom Description	Symb #	Stmnt Cnt	BEnd lvl	Proc lvl	List mark	id mark	Var mark	Var Scpe	Read Wrte	Decl Loca	Import. OpExport Identifier	Type	Values
1274	[145	119	4	1	238	0	7	-	-	0	-	-	-	-
1275	zone	1	119	4	1	238	500	7	P	-	472	-	-	-	-
1276	,	9	119	4	1	238	0	7	-	-	0	-	-	-	-
1277	tempString	1	119	4	1	238	500	7	L	-	966	-	-	-	-
1278]	139	119	4	1	0	0	7	-	-	0	-	-	-	-
1279	;	10	120	4	1	0	0	7	-	-	0	-	-	-	-
1280	log	1	120	4	1	0	237	7	-	-	706	-	-	-	-
1281	[145	120	4	1	238	0	7	-	-	0	-	-	-	-
1282	"	6	120	4	1	238	0	7	-	-	0	-	-	-	-
1283]	139	120	4	1	0	0	7	-	-	0	-	-	-	-
1284	}	140	120	3	1	0	0	7	-	-	0	-	-	-	-
1285	ELSE	106	120	3	1	0	0	7	-	-	0	-	-	-	-
1286	{	146	120	4	1	0	0	7	-	-	0	-	-	-	-
1287	fileStackPtr	0	120	4	1	0	500	7	G	R	432	-	-	-	-
1288	[145	120	4	1	201	0	7	-	-	0	-	-	-	-
1289	fileCount	1	120	4	1	201	500	7	G	-	438	-	-	-	-
1290]	139	120	4	1	0	0	7	-	-	0	-	-	-	-
1291	,	27	120	4	1	0	0	7	-	-	0	-	-	-	-
1292	fileCount	1	120	4	1	0	500	7	D	W	1287	-	fileStackPtr	-	-
1293		14	120	4	1	0	0	7	-	-	0	-	-	-	-
1294	fileCount	1	120	4	1	0	500	7	G	R	438	-	-	-	-
1295	;	10	121	4	1	0	0	7	-	-	0	-	-	-	-
1296	String	0	121	4	1	0	0	7	-	R	0	-	-	-	-
1297	.	27	121	4	1	0	0	7	-	-	0	-	-	-	-
1298	FreeString	1	121	4	1	0	500	7	M	R	1296	-	String	-	-
1299	[145	121	4	1	201	0	7	-	-	0	-	-	-	-
1300	zone	1	121	4	1	201	500	7	P	-	472	-	-	-	-
1301	,	9	121	4	1	201	0	7	-	-	0	-	-	-	-
1302	fileStackPtr	0	121	4	1	201	500	7	G	-	432	-	-	-	-
1303	[145	121	4	1	201	0	7	-	-	0	-	-	-	-
1304	fileCount	1	121	4	1	201	500	7	G	-	438	-	-	-	-
1305]	139	121	4	1	0	0	7	-	-	0	-	-	-	-
1306	.	27	121	4	1	0	0	7	-	-	0	-	-	-	-
1307	fileName	1	121	4	1	0	500	7	D	R	1302	-	fileStackPtr	-	-
1308]	139	121	4	1	0	0	7	-	-	0	-	-	-	-
1309	;	10	122	4	1	0	0	7	-	-	0	-	-	-	-
1310	fileStackPtr	0	122	4	1	0	500	7	G	R	432	-	-	-	-
1311	[145	122	4	1	201	0	7	-	-	0	-	-	-	-
1312	fileCount	1	122	4	1	201	500	7	G	-	438	-	-	-	-
1313]	139	122	4	1	0	0	7	-	-	0	-	-	-	-
1314	.	27	122	4	1	0	0	7	-	-	0	-	-	-	-
1315	fileName	1	122	4	1	0	500	7	D	W	1310	-	fileStackPtr	-	-
1316		14	122	4	1	0	0	7	-	-	0	-	-	-	-
1317	String	0	122	4	1	0	0	7	-	R	0	-	-	-	-
1318	,	27	122	4	1	0	0	7	-	-	0	-	-	-	-
1319	CopyToNewString	1	122	4	1	0	237	7	M	-	1317	-	String	-	-
1320	[145	122	4	1	238	0	7	-	-	0	-	-	-	-
1321	name	1	122	4	1	238	500	7	M	-	0	-	-	-	-
1322	,	9	122	4	1	238	0	7	-	-	0	-	-	-	-

DESIGN EXTRACTION STATISTICS
=====

Ind #	Atom Description	Symb #	Stmt Cnt	BEnd lvl	Proc lvl	List mark	Var id mark	Var Scpe	Read Wrte	Decl Loca	Import OpExport Identifier	Type	Values
1323	zone	1	122	4	1	1	238	500	7	P	-	472	-	-
1324]	139	122	4	1	1	0	0	7	-	-	0	-	-
1325	;	10	123	4	1	1	0	0	7	-	-	0	-	-
1326	fileStackPtr	0	123	4	1	1	0	500	7	G	R	432	-	-
1327	[145	123	4	1	1	201	0	7	-	-	0	-	-
1328	fileCount	1	123	4	1	1	201	500	7	G	-	438	-	-
1329]	139	123	4	1	1	0	0	7	-	-	0	-	-
1330		27	123	4	1	1	0	0	7	-	-	0	-	-
1331	startIndex	1	123	4	1	1	0	500	7	D	W	1326	fileStackPtr	-
1332		14	123	4	1	1	0	0	7	-	-	0	-	-
1333	fileStackPtr	0	123	4	1	1	0	500	7	G	R	432	-	-
1334	[145	123	4	1	1	201	0	7	-	-	0	-	-
1335	fileCount	1	123	4	1	1	201	500	7	G	-	438	-	-
1336	-	23	123	4	1	1	201	0	7	-	-	0	-	-
1337	1	2	123	4	1	1	201	0	7	-	-	0	-	-
1338]	139	123	4	1	1	0	0	7	-	-	0	-	-
1339	.	27	123	4	1	1	0	0	7	-	-	0	-	-
1340	endIndex	1	123	4	1	1	0	500	7	D	R	1333	fileStackPtr	-
1341	+	22	123	4	1	1	0	0	7	-	-	0	-	-
1342	1	2	123	4	1	1	0	0	7	-	-	0	-	-
1343	;	10	124	4	1	1	0	0	7	-	-	0	-	-
1344	Stream	0	124	4	1	1	0	0	7	-	-	0	-	-
1345	.	27	124	4	1	1	0	0	7	-	-	0	-	-
1346	setPosition	1	124	4	1	1	0	237	7	M	-	1344	Stream	-
1347	[145	124	4	1	1	238	0	7	-	-	0	-	-
1348	SH	1	124	4	1	1	238	501	7	-	-	0	file	-
1349	;	11	124	4	1	1	238	0	7	-	-	0	-	-
1350	file	0	124	4	1	1	238	0	7	-	-	0	-	-
1351	,	9	124	4	1	1	238	0	7	-	-	0	-	-
1352	position	1	124	4	1	1	238	238	7	-	-	0	-	-
1353	;	11	124	4	1	1	238	0	7	-	-	0	-	-
1354	0	2	124	4	1	1	238	0	7	-	-	0	-	-
1355]	139	124	4	1	1	0	0	7	-	-	0	-	-
1356	;	10	125	4	1	1	0	0	7	-	-	0	-	-
1357	parseZone	1	125	4	1	1	0	500	7	L	W	959	-	-
1358		14	125	4	1	1	0	0	7	-	-	0	-	-
1359	Heap	0	125	4	1	1	0	0	7	-	-	0	-	-
1360	.	27	125	4	1	1	0	0	7	-	-	0	-	-
1361	Create	1	125	4	1	1	0	237	7	M	-	1359	Heap	-
1362	[145	125	4	1	1	238	0	7	-	-	0	-	-
1363	checking	1	125	4	1	1	238	238	7	-	-	0	-	-
1364	;	11	125	4	1	1	238	0	7	-	-	0	-	-
1365	TRUE	204	125	4	1	1	238	0	7	-	-	0	-	-
1366	,	9	125	4	1	1	238	0	7	-	-	0	-	-
1367	ownerChecking	1	125	4	1	1	238	238	7	-	-	0	-	-
1368	;	11	125	4	1	1	238	0	7	-	-	0	-	-
1369	TRUE	204	125	4	1	1	238	0	7	-	-	0	-	-
1370	,	9	125	4	1	1	238	0	7	-	-	0	-	-
1371	initial	1	125	4	1	1	238	238	7	-	-	0	-	-

EXTRACTOR.DATA

CUT SHORT AT THIS POINT

TO SAVE PAPER

APPENDIX A - OUTPUT LISTINGS

A.2

OUTPUT LISTING OF STRUCTCHART.DATA

A.2.1 SCGRAPH

DesignExtractor 1.0 - By: David Egerton

Mesa Source Files; DesignExtractorImpl

ParserImpl DisplayImpl StructChartImpl

Parse! Display! Clear!

SC Graph

DFD Variable

Raw Data

SC Table

DFD Table

Repeats

Analysis Complete. 4 files.

~ ~ ~ Processing Display ~ ~ ~

Parsed file statistics written... ..

Missing export information written.....

Creating Structure Chart....

- Studying repeat calls.....
- Creating Semi Graphical Output.
- Creating Tabular Output.....

Done .. Output created to file StructChart.data

~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

Updating DataBase with DataFlow Info

~ ~ ~ Display Complete ~ ~ ~ ~ ~

DESIGN EXTRACTION STATISTICS
=====

The files parsed in this session were:

Index	File Name	Start Index	End Index	Exports to
1	DesignExtractorImpl.mesa	1	3778	Utils
2	DisplayImpl.mesa	3779	3945	Display
3	ParserImpl.mesa	3946	5725	Parser
4	StructChartImpl.mesa	5726	12305	Display

STRUCTURE CHART - TEXTURAL OUTPUT TABLE - MISSING EXPORT INFORMATION

The following is the list of unknown export information. The files loaded in this session do not contain the exported procedures that follow:-

LINE #	MISSING EXPORT	CALL MADE FROM FILE	PROCEDURE CALLED
1	Put	DesignExtractorImpl.mesa	Text
2	ToolWindow	DesignExtractorImpl.mesa	Activate
3	Exec	DesignExtractorImpl.mesa	OutputProc
4	Tool	DesignExtractorImpl.mesa	Destroy
5	Exec	DesignExtractorImpl.mesa	RemoveCommand
6	Heap	DesignExtractorImpl.mesa	Delete
7	UserInput	DesignExtractorImpl.mesa	UserAbort
8	file	DesignExtractorImpl.mesa	delete
9	Utils	DesignExtractorImpl.mesa	CardToString
10	MStream	DesignExtractorImpl.mesa	Create
11	Format	DesignExtractorImpl.mesa	Line
12	String	DesignExtractorImpl.mesa	CopyToNewString
13	String	DesignExtractorImpl.mesa	FreeString
14	Stream	DesignExtractorImpl.mesa	SetPosition
15	Parser	DesignExtractorImpl.mesa	ScanInit
16	Utils	DesignExtractorImpl.mesa	ClearMetFile
17	Process	DesignExtractorImpl.mesa	Detach
18	Process	DesignExtractorImpl.mesa	SetPriority
19	Process	DesignExtractorImpl.mesa	Yield
20	Utils	DesignExtractorImpl.mesa	OpenMetFile
21	Utils	DesignExtractorImpl.mesa	MetFileLength
22	Utils	DesignExtractorImpl.mesa	CloseMetFile
23	token	DesignExtractorImpl.mesa	FreeStringHandle
24	token	DesignExtractorImpl.mesa	MaybeQuoted
25	String	DesignExtractorImpl.mesa	Empty
26	Token	DesignExtractorImpl.mesa	FreeTokenString
27	MFile	DesignExtractorImpl.mesa	EnumeratedDirectory
28	Format	DesignExtractorImpl.mesa	LongNumber
29	TextSW	DesignExtractorImpl.mesa	ForceOutput
30	String	DesignExtractorImpl.mesa	EquivalentSubStrings
31	String	DesignExtractorImpl.mesa	AppendStringAndGrow
32	fileList	DesignExtractorImpl.mesa	Add
33	ChangeTable	DesignExtractorImpl.mesa	Init
34	ChangeTable	DesignExtractorImpl.mesa	MakeTable
35	fileList	DesignExtractorImpl.mesa	AllEntryNames
36	ChangeTable	DesignExtractorImpl.mesa	CleanUp
37	FormSW	DesignExtractorImpl.mesa	AllocateItemDescriptor
38	FormSW	DesignExtractorImpl.mesa	StringItem
39	FormSW	DesignExtractorImpl.mesa	BooleanItem
40	FormSW	DesignExtractorImpl.mesa	CommandItem
41	Tool	DesignExtractorImpl.mesa	MakeFormSW
42	Tool	DesignExtractorImpl.mesa	UnusedLogName
43	Tool	DesignExtractorImpl.mesa	MakeFileSW
44	ToolDriver	DesignExtractorImpl.mesa	NoteSWs
45	HeraldWindow	DesignExtractorImpl.mesa	AppendMessage
46	Exec	DesignExtractorImpl.mesa	AddCommand
47	String	DesignExtractorImpl.mesa	Copy
48	MSegment	DesignExtractorImpl.mesa	Address
49	Runtime	DesignExtractorImpl.mesa	GetTableBase
50	Display	DisplayImpl.mesa	RawData

STRUCTURE CHART - TEXTURAL OUTPUT TABLE - MISSING EXPORT INFORMATION

=====			
The following is the list of unknown export information. The files loaded in this session do not contain the exported procedures that follow:-			
LINE #	MISSING EXPORT	CALL MADE FROM FILE	PROCEDURE CALLED
51	Parser	ParserImpl.mesa	ScanReset
52	Utils	ParserImpl.mesa	StringEqual
53	Parser	ParserImpl.mesa	processQueue
54	Parser	ParserImpl.mesa	BaseTable
55	String	ParserImpl.mesa	Equal
56	Parser	ParserImpl.mesa	pass2
57	Parser	ParserImpl.mesa	pass3
58	Parser	ParserImpl.mesa	pass4
59	Parser	ParserImpl.mesa	pass5
60	Parser	ParserImpl.mesa	StateSeq
61	Parser	ParserImpl.mesa	LinkSeq
62	Parser	ParserImpl.mesa	ValueSeq
63	Parser	ParserImpl.mesa	AssignDescriptors
64	Parser	ParserImpl.mesa	ActionSeq
65	Display	StructChartImpl.mesa	DataFlow
66	MStream	StructChartImpl.mesa	SetLength
67	Stream	StructChartImpl.mesa	GetPosition
68	Stream	StructChartImpl.mesa	PutChar
69	Stream	StructChartImpl.mesa	PutString
70	String	StructChartImpl.mesa	Length
71	String	StructChartImpl.mesa	MakeString
72	String	StructChartImpl.mesa	AppendSubString

STRUCTURE CHART - SEMI GRAPHICAL OUTPUT TABLE

STRUCTURE CHART - SEMI GRAPHICAL OUTPUT TABLE																	
=====																	
Line	Called From Filename	Interface Name	Declared in Filename	Lvl	SLC	1	2	3	4	5	6	7	8	9	10	11	12
1	DesignExtractorImpl.	FormSW	-	1	SLC	GenTextDisplay											
2	DesignExtractorImpl.	-	DesignExtractorImpl.	2		Enter											
3	DesignExtractorImpl.	-	DesignExtractorImpl.	2		Exit											
4	DesignExtractorImpl.	Heap	-	2		Delete											
5	DesignExtractorImpl.	Heap	-	2		Create											
6	DesignExtractorImpl.	Process	-	2		Detach											
7	DesignExtractorImpl.	-	DesignExtractorImpl.	2		TextDisplay											
8	DesignExtractorImpl.	Heap	-	3		Delete											
9	DesignExtractorImpl.	-	DesignExtractorImpl.	3		Exit											
10	DesignExtractorImpl.	Process	-	3		SetPriority											
11	DesignExtractorImpl.	Process	-	3		Yield											
12	DesignExtractorImpl.	-	DesignExtractorImpl.	3		log											
13	DesignExtractorImpl.	Put	-	4		Text											
14	DesignExtractorImpl.	-	DesignExtractorImpl.	3		CheckSelection											
15	DesignExtractorImpl.	-	DesignExtractorImpl.	4		log											
16	DesignExtractorImpl.	Put	-	5		Text											
31	DesignExtractorImpl.	-	DesignExtractorImpl.	3		CheckForAbort											
32	DesignExtractorImpl.	UserInput	-	4		UserAbort											
33	DesignExtractorImpl.	Display	-	3		Control											
34	DisplayImpl.mesa	Display	-	4		RawData											
35	DisplayImpl.mesa	Display	-	4		StructChart											
36	StructChartImpl.mesa	Utils	-	5		CheckForAbort											
37	DesignExtractorImpl.	UserInput	-	6		UserAbort											
38	StructChartImpl.mesa	MStream	-	5		WriteOnly											
39	StructChartImpl.mesa	statsFile	-	5		delete											
40	StructChartImpl.mesa	-	StructChartImpl.mesa	5		FormatData											
41	StructChartImpl.mesa	-	StructChartImpl.mesa	6		WriteFiles											
42	StructChartImpl.mesa	-	StructChartImpl.mesa	7		StatsWrite											
43	StructChartImpl.mesa	Stream	-	8		PutString											
44	StructChartImpl.mesa	Stream	-	7		PutChar											
61	StructChartImpl.mesa	-	StructChartImpl.mesa	7		CheckLength											
62	StructChartImpl.mesa	String	-	8		Length											
63	StructChartImpl.mesa	-	StructChartImpl.mesa	8		AddSpaces											
64	StructChartImpl.mesa	String	-	9		AppendStringAndGrow											
65	StructChartImpl.mesa	-	StructChartImpl.mesa	8		ReduceLength											
66	StructChartImpl.mesa	String	-	9		MakeString											
67	StructChartImpl.mesa	String	-	9		AppendSubString											
68	StructChartImpl.mesa	String	-	9		FreeString											
69	StructChartImpl.mesa	String	-	9		CopyToNewString											
73	StructChartImpl.mesa	String	-	7		FreeString											
136	StructChartImpl.mesa	Utils	-	6		log											
137	DesignExtractorImpl.	Put	-	7		Text											
138	StructChartImpl.mesa	-	StructChartImpl.mesa	6		WriteExportInfo											
139	StructChartImpl.mesa	-	StructChartImpl.mesa	7		CheckProcExists											
140	StructChartImpl.mesa	String	-	7		FreeString											
141	StructChartImpl.mesa	-	StructChartImpl.mesa	7		OutputExportInfo											
142	StructChartImpl.mesa	-	StructChartImpl.mesa	8		WriteExportTitles											
143	StructChartImpl.mesa	-	StructChartImpl.mesa	9		StatsWrite											
144	StructChartImpl.mesa	Stream	-	10		PutString											
145	StructChartImpl.mesa	Stream	-	9		CopyToNewString											
186	StructChartImpl.mesa	String	-	8		CardToString											
187	StructChartImpl.mesa	Utils	-	8		CheckLength											
188	StructChartImpl.mesa	-	StructChartImpl.mesa	8													

STRUCTURE CHART - SEMI GRAPHICAL OUTPUT TABLE

Line	Called From Filename	Interface Name	Declared in Filename	Lvl	SLC	1	2	3	4	5	6	7	8	9	10	11	12
189	StructChartImpl.mesa	String	-	9											Length		
190	StructChartImpl.mesa	-	StructChartImpl.mesa	9											AddSpaces		
191	StructChartImpl.mesa	String	-	10													
AppendStringAndGrow																	
192	StructChartImpl.mesa	-	StructChartImpl.mesa	9											ReduceLength		
193	StructChartImpl.mesa	String	-	10											MakeString		
194	StructChartImpl.mesa	String	-	10											AppendSubString		
195	StructChartImpl.mesa	String	-	10											FreeString		
196	StructChartImpl.mesa	String	-	10											CopyToNewString		
198	StructChartImpl.mesa	-	StructChartImpl.mesa	8											StatsWrite		
199	StructChartImpl.mesa	Stream	-	9											PutString		
200	StructChartImpl.mesa	String	-	8											FreeString		
203	StructChartImpl.mesa	String	-	8											Equal		
249	StructChartImpl.mesa	Stream	-	8											PutChar		
255	StructChartImpl.mesa	Utils	-	6											CheckForAbort		
256	DesignExtractorImpl.	UserInput	-	7											UserAbort		
257	StructChartImpl.mesa	-	StructChartImpl.mesa	6											CheckReturnCondition		
258	StructChartImpl.mesa	-	StructChartImpl.mesa	7											StatsWrite		
259	StructChartImpl.mesa	Stream	-	8											PutString		
260	StructChartImpl.mesa	String	-	7											CopyToNewString		
261	StructChartImpl.mesa	Utils	-	7											CardToString		
262	StructChartImpl.mesa	-	StructChartImpl.mesa	7											CheckLength		
263	StructChartImpl.mesa	String	-	8											Length		
264	StructChartImpl.mesa	-	StructChartImpl.mesa	8											AddSpaces		
265	StructChartImpl.mesa	String	-	9											AppendStringAndGrow		
266	StructChartImpl.mesa	-	StructChartImpl.mesa	8											ReduceLength		
267	StructChartImpl.mesa	String	-	9											MakeString		
268	StructChartImpl.mesa	String	-	9											AppendSubString		
269	StructChartImpl.mesa	String	-	9											FreeString		
270	StructChartImpl.mesa	String	-	9											CopyToNewString		
274	StructChartImpl.mesa	String	-	7											FreeString		
275	StructChartImpl.mesa	Stream	-	7											PutChar		
312	StructChartImpl.mesa	-	StructChartImpl.mesa	6											CheckIfTerminalDecl		
313	StructChartImpl.mesa	-	StructChartImpl.mesa	6											CleanUpPositionData		
314	StructChartImpl.mesa	-	StructChartImpl.mesa	6											CollectPositionData		
315	StructChartImpl.mesa	-	StructChartImpl.mesa	7											StatsWrite		
316	StructChartImpl.mesa	Stream	-	8											PutString		
319	StructChartImpl.mesa	String	-	7											FreeString		
335	StructChartImpl.mesa	Stream	-	7											PutChar		
336	StructChartImpl.mesa	-	StructChartImpl.mesa	6											UpdateFullStack		
337	StructChartImpl.mesa	Utils	-	7											CardToString		
338	StructChartImpl.mesa	Utils	-	7											log		
339	DesignExtractorImpl.	Put	-	8											Text		
344	StructChartImpl.mesa	-	StructChartImpl.mesa	7											FindFileNumber		
345	StructChartImpl.mesa	-	StructChartImpl.mesa	6											SuppressOutput		
346	StructChartImpl.mesa	-	StructChartImpl.mesa	7											StatsWrite		
347	StructChartImpl.mesa	Stream	-	8											PutString		
348	StructChartImpl.mesa	String	-	7											CopyToNewString		
349	StructChartImpl.mesa	Utils	-	7											CardToString		
350	StructChartImpl.mesa	-	StructChartImpl.mesa	7											CheckLength		
351	StructChartImpl.mesa	String	-	8											Length		
352	StructChartImpl.mesa	-	StructChartImpl.mesa	8											AddSpaces		
353	StructChartImpl.mesa	String	-	9											AppendStringAndGrow		

354 StructChartImpl.mesa - StructChartImpl.mesa 8

STRUCTURE CHART - SEMI GRAPHICAL OUTPUT TABLE

STRUCTURE CHART - SEMI GRAPHICAL OUTPUT TABLE																	

Line	Called From Filename	Interface Name	Declared in Filename	Lvl	SLC	1	2	3	4	5	6	7	8	9	10	11	12
-----	-----	-----	-----	---	---	---	---	---	---	---	---	---	---	---	---	---	---
355	StructChartImpl.mesa	String	-	9												MakeString	
356	StructChartImpl.mesa	String	-	9												AppendSubString	
357	StructChartImpl.mesa	String	-	9												FreeString	
358	StructChartImpl.mesa	String	-	9												CopyToNewString	
362	StructChartImpl.mesa	String	-	7									FreeString				
382	StructChartImpl.mesa	Stream	-	7									PutChar				
453	StructChartImpl.mesa	-	StructChartImpl.mesa	6									SearchForDeclaration				
454	StructChartImpl.mesa	-	StructChartImpl.mesa	7									CheckProcExists				
455	StructChartImpl.mesa	String	-	7									FreeString				
456	StructChartImpl.mesa	-	StructChartImpl.mesa	7									FindFileName				
488	StructChartImpl.mesa	-	StructChartImpl.mesa	6									GoFindCallParams				
491	StructChartImpl.mesa	-	StructChartImpl.mesa	6									StudyRepeats				
494	StructChartImpl.mesa	-	StructChartImpl.mesa	6									WriteSCTitles				
495	StructChartImpl.mesa	Stream	-	7									PutChar				
496	StructChartImpl.mesa	-	StructChartImpl.mesa	7									StatsWrite				
497	StructChartImpl.mesa	Stream	-	8									PutString				
508	StructChartImpl.mesa	-	StructChartImpl.mesa	6									SemiGraphOutput				
509	StructChartImpl.mesa	String	-	7									CopyToNewString				
510	StructChartImpl.mesa	Utils	-	7									CardToString				
511	StructChartImpl.mesa	-	StructChartImpl.mesa	7									CheckLength				
512	StructChartImpl.mesa	String	-	8									Length				
513	StructChartImpl.mesa	-	StructChartImpl.mesa	8									AddSpaces				
514	StructChartImpl.mesa	String	-	9									ReduceLength				
515	StructChartImpl.mesa	-	StructChartImpl.mesa	8									AppendStringAndGrow				
516	StructChartImpl.mesa	String	-	9									MakeString				
517	StructChartImpl.mesa	String	-	9									AppendSubString				
518	StructChartImpl.mesa	String	-	9									FreeString				
519	StructChartImpl.mesa	String	-	9									CopyToNewString				
521	StructChartImpl.mesa	-	StructChartImpl.mesa	7									StatsWrite				
522	StructChartImpl.mesa	Stream	-	8									PutString				
523	StructChartImpl.mesa	String	-	7									FreeString				
557	StructChartImpl.mesa	-	StructChartImpl.mesa	7									FindFileName				
589	StructChartImpl.mesa	Stream	-	7									PutChar				
590	StructChartImpl.mesa	Stream	-	7									WriteSCTitles				
591	StructChartImpl.mesa	Stream	-	8									PutChar				
592	StructChartImpl.mesa	-	StructChartImpl.mesa	8									StatsWrite				
593	StructChartImpl.mesa	Stream	-	9									PutString				
607	StructChartImpl.mesa	-	StructChartImpl.mesa	6									TabularOutput				
608	StructChartImpl.mesa	-	StructChartImpl.mesa	7									WriteTableTitles				
609	StructChartImpl.mesa	Stream	-	8									PutChar				
611	StructChartImpl.mesa	-	StructChartImpl.mesa	8									StatsWrite				
612	StructChartImpl.mesa	Stream	-	9									PutString				
629	StructChartImpl.mesa	-	StructChartImpl.mesa	7									OutputTableLine				
630	StructChartImpl.mesa	-	StructChartImpl.mesa	8									StatsWrite				
631	StructChartImpl.mesa	Stream	-	9									PutString				
632	StructChartImpl.mesa	String	-	8									CopyToNewString				
633	StructChartImpl.mesa	Utils	-	8									CardToString				
634	StructChartImpl.mesa	-	StructChartImpl.mesa	8									CheckLength				
635	StructChartImpl.mesa	String	-	9									Length				
636	StructChartImpl.mesa	-	StructChartImpl.mesa	9									AddSpaces				
637	StructChartImpl.mesa	String	-	10													
AppendStringAndGrow																	

StructChart.data 4-Feb-94 18:54:45 EST

638 StructChartImpl.mesa -
639 StructChartImpl.mesa String

9
10

StructChartImpl.mesa
-

ReduceLength
MakeString

STRUCTURE CHART - SEMI GRAPHICAL OUTPUT TABLE

Indented Procedure Call Levels

Line	Called From Filename	Interface Name	Declared in Filename	Lvl	SLC	1	2	3	4	5	6	7	8	9	10	11	12
640	StructChartImpl.mesa	String	-	10													
641	StructChartImpl.mesa	String	-	10													
642	StructChartImpl.mesa	String	-	10													
646	StructChartImpl.mesa	String	-	8													
649	StructChartImpl.mesa	-	StructChartImpl.mesa	8													
650	StructChartImpl.mesa	String	-	9													
651	StructChartImpl.mesa	String	-	9													
654	StructChartImpl.mesa	-	StructChartImpl.mesa	9													
655	StructChartImpl.mesa	String	-	10													
656	StructChartImpl.mesa	-	StructChartImpl.mesa	10													
657	StructChartImpl.mesa	String	-	11													
AppendStringAndGrow																	
658	StructChartImpl.mesa	-	StructChartImpl.mesa	10													
659	StructChartImpl.mesa	String	-	11													
660	StructChartImpl.mesa	String	-	11													
AppendSubString																	
661	StructChartImpl.mesa	String	-	11													
662	StructChartImpl.mesa	String	-	11													
CopyToNewString																	
664	StructChartImpl.mesa	-	StructChartImpl.mesa	9													
665	StructChartImpl.mesa	Stream	-	10													
666	StructChartImpl.mesa	String	-	9													
699	StructChartImpl.mesa	-	StructChartImpl.mesa	8													
700	StructChartImpl.mesa	String	-	8													
701	StructChartImpl.mesa	-	StructChartImpl.mesa	8													
702	StructChartImpl.mesa	String	-	9													
703	StructChartImpl.mesa	String	-	9													
704	StructChartImpl.mesa	String	-	9													
705	StructChartImpl.mesa	String	-	9													
720	StructChartImpl.mesa	Stream	-	8													
721	StructChartImpl.mesa	-	StructChartImpl.mesa	8													
722	StructChartImpl.mesa	Stream	-	9													
724	StructChartImpl.mesa	-	StructChartImpl.mesa	9													
725	StructChartImpl.mesa	Stream	-	10													
1340	StructChartImpl.mesa	Stream	-	5													
1341	StructChartImpl.mesa	Utils	-	5													
1342	DesignExtractorImpl.	Put	-	6													
1345	StructChartImpl.mesa	Display	-	5													
1346	StructChartImpl.mesa	MStream	-	5													
1347	StructChartImpl.mesa	Stream	-	5													
1355	DesignExtractorImpl.	FormSW	-	1	SLC												
1356	DesignExtractorImpl.	-	DesignExtractorImpl.	2													
1357	DesignExtractorImpl.	-	DesignExtractorImpl.	2													
1358	DesignExtractorImpl.	Process	-	2													
1359	DesignExtractorImpl.	-	DesignExtractorImpl.	2													
1360	DesignExtractorImpl.	-	DesignExtractorImpl.	3													
1361	DesignExtractorImpl.	Process	-	3													
1362	DesignExtractorImpl.	Process	-	3													
1363	DesignExtractorImpl.	-	DesignExtractorImpl.	3													
1364	DesignExtractorImpl.	Put	-	4													
1367	DesignExtractorImpl.	-	DesignExtractorImpl.	3													

STRUCTURE CHART -- SEMI GRAPHICAL OUTPUT TABLE

Line	Called From Filename	Interface Name	Declared in Filename	Lvl	SLC	1	2	3	4	5	6	7	8	9	10	11	12
1374	DesignExtractorImpl.	Heap	-	3					Delete								
1375	DesignExtractorImpl.	-	DesignExtractorImpl.	3					CheckForAbort								
1376	DesignExtractorImpl.	UserInput	-	4					UserAbort								
1377	DesignExtractorImpl.	Utils	-	3					ClearMetFile								
1381	DesignExtractorImpl.	FormSW	-	1	SLC	Run											
1382	DesignExtractorImpl.	-	DesignExtractorImpl.	2			Enter										
1383	DesignExtractorImpl.	-	DesignExtractorImpl.	2			Exit										
1384	DesignExtractorImpl.	Process	-	2			Detach										
1385	DesignExtractorImpl.	-	DesignExtractorImpl.	2			RunIt										
1386	DesignExtractorImpl.	-	DesignExtractorImpl.	3				Exit									
1387	DesignExtractorImpl.	Process	-	3				SetPriority									
1388	DesignExtractorImpl.	Process	-	3				Yield									
1389	DesignExtractorImpl.	-	DesignExtractorImpl.	3				log									
1390	DesignExtractorImpl.	Put	-	4					Text								
1395	DesignExtractorImpl.	Heap	-	3					Create								
1396	DesignExtractorImpl.	Heap	-	3					Delete								
1397	DesignExtractorImpl.	Utils	-	3					OpenMetFile								
1400	DesignExtractorImpl.	Utils	-	3					MetFileLength								
1401	DesignExtractorImpl.	Utils	-	3					CloseMetFile								
1402	DesignExtractorImpl.	token	-	3					FreeStringHandle								
1403	DesignExtractorImpl.	token	-	3					MaybeQuoted								
1404	DesignExtractorImpl.	String	-	3					Empty								
1405	DesignExtractorImpl.	Token	-	3					FreeTokenString								
1406	DesignExtractorImpl.	-	DesignExtractorImpl.	3					AppendExtensionIfNeeded								
1407	DesignExtractorImpl.	String	-	4					EquivalentSubStrings								
1408	DesignExtractorImpl.	-	-	4					to								
1409	DesignExtractorImpl.	String	-	4					AppendStringAndGrow								
1410	DesignExtractorImpl.	MFile	-	3					EnumeratedDirectory								
1411	DesignExtractorImpl.	-	DesignExtractorImpl.	3					ProcessFile								
1412	DesignExtractorImpl.	-	DesignExtractorImpl.	4					log								
1413	DesignExtractorImpl.	Put	-	5					Text								
1414	DesignExtractorImpl.	-	DesignExtractorImpl.	4					Cleanup								
1415	DesignExtractorImpl.	file	-	5					delete								
1416	DesignExtractorImpl.	-	DesignExtractorImpl.	5					log								
1417	DesignExtractorImpl.	Put	-	6					Text								
1424	DesignExtractorImpl.	Utils	-	5					CardToString								
1431	DesignExtractorImpl.	-	DesignExtractorImpl.	4					CheckForAbort								
1432	DesignExtractorImpl.	UserInput	-	5					UserAbort								
1433	DesignExtractorImpl.	MStream	-	4					Create								
1436	DesignExtractorImpl.	Format	-	4					Line								
1445	DesignExtractorImpl.	String	-	4					CopyToNewString								
1446	DesignExtractorImpl.	Utils	-	4					CardToString								
1449	DesignExtractorImpl.	String	-	4					FreeString								
1453	DesignExtractorImpl.	Stream	-	4					SetPosition								
1455	DesignExtractorImpl.	Heap	-	4					Delete								
1456	DesignExtractorImpl.	Parser	-	4					ScanInit								
1457	DesignExtractorImpl.	Parser	-	4					Parse								
1458	ParserImpl.mesa	-	ParserImpl.mesa	5					ParseInit								

APPENDIX A - OUTPUT LISTINGS

A.2

OUTPUT LISTING OF STRUCTCHART.DATA

A.2.2 SCTABLE

DESIGN EXTRACTION STATISTICS
=====

The files parsed in this session were:

Index	File Name	Start Index	End Index	Exports to
1	DesignExtractorImpl.mesa	1	3778	Utils
2	DisplayImpl.mesa	3779	3946	Display
3	ParserImpl.mesa	3946	5726	Parser
4	StructChartImpl.mesa	5726	12306	Display

STRUCTURE CHART - TEXTURAL OUTPUT TABLE - MISSING EXPORT INFORMATION

The following is the list of unknown export information. The files loaded in this session do not contain the exported procedures that follow:-

LINE #	MISSING EXPORT	CALL MADE FROM FILE	PROCEDURE CALLED
1	Put	DesignExtractorImpl.mesa	Text
2	ToolWindow	DesignExtractorImpl.mesa	Activate
3	Exec	DesignExtractorImpl.mesa	OutputProc
4	Tool	DesignExtractorImpl.mesa	Destroy
5	Exec	DesignExtractorImpl.mesa	RemoveCommand
6	Heap	DesignExtractorImpl.mesa	Delete
7	UserInput	DesignExtractorImpl.mesa	UserAbort
8	file	DesignExtractorImpl.mesa	delete
9	Utils	DesignExtractorImpl.mesa	CardToString
10	MStream	DesignExtractorImpl.mesa	Create
11	Format	DesignExtractorImpl.mesa	Line
12	String	DesignExtractorImpl.mesa	CopyToNewString
13	String	DesignExtractorImpl.mesa	FreeString
14	Stream	DesignExtractorImpl.mesa	SetPosition
15	Parser	DesignExtractorImpl.mesa	ScanInit
16	Utils	DesignExtractorImpl.mesa	CleanMetFile
17	Process	DesignExtractorImpl.mesa	Detach
18	Process	DesignExtractorImpl.mesa	SetPriority
19	Process	DesignExtractorImpl.mesa	Yield
20	Utils	DesignExtractorImpl.mesa	OpenMetFile
21	Utils	DesignExtractorImpl.mesa	MetFileLength
22	Utils	DesignExtractorImpl.mesa	CloseMetFile
23	token	DesignExtractorImpl.mesa	FreeStringHandle
24	token	DesignExtractorImpl.mesa	MaybeQuoted
25	String	DesignExtractorImpl.mesa	Empty
26	Token	DesignExtractorImpl.mesa	FreeTokenString
27	MFile	DesignExtractorImpl.mesa	EnumeratedDirectory
28	Format	DesignExtractorImpl.mesa	LongNumber
29	TextSW	DesignExtractorImpl.mesa	ForceOutput
30	String	DesignExtractorImpl.mesa	EquivalentSubStrings
31	String	DesignExtractorImpl.mesa	AppendStringAndGrow
32	fileList	DesignExtractorImpl.mesa	Add
33	ChangeTable	DesignExtractorImpl.mesa	Init
34	ChangeTable	DesignExtractorImpl.mesa	MakeTable
35	fileList	DesignExtractorImpl.mesa	AllEntryNames
36	ChangeTable	DesignExtractorImpl.mesa	Cleanup
37	FormSW	DesignExtractorImpl.mesa	AllocateItemDescriptor
38	FormSW	DesignExtractorImpl.mesa	StringItem
39	FormSW	DesignExtractorImpl.mesa	BooleanItem
40	FormSW	DesignExtractorImpl.mesa	CommandItem
41	Tool	DesignExtractorImpl.mesa	MakeFormSW
42	Tool	DesignExtractorImpl.mesa	UnusedLogName
43	Tool	DesignExtractorImpl.mesa	MakeFileSW
44	ToolDriver	DesignExtractorImpl.mesa	NotesWS
45	HeraldWindow	DesignExtractorImpl.mesa	AppendMessage
46	Exec	DesignExtractorImpl.mesa	AddCommand
47	String	DesignExtractorImpl.mesa	Copy
48	MSegment	DesignExtractorImpl.mesa	Address
49	Runtime	DesignExtractorImpl.mesa	GetTableBase
50	Display	DisplayImpl.mesa	RawData

STRUCTURE CHART - TEXTURAL OUTPUT TABLE - MISSING EXPORT INFORMATION

=====		
The following is the list of unknown export information. The files loaded in this session do not contain the exported procedures that follow:-		

LINE #	MISSING EXPORT	CALL MADE FROM FILE

51	Parser	ParserImpl.mesa
52	Utils	ParserImpl.mesa
53	Parser	ParserImpl.mesa
54	Parser	ParserImpl.mesa
55	String	ParserImpl.mesa
56	Parser	ParserImpl.mesa
57	Parser	ParserImpl.mesa
58	Parser	ParserImpl.mesa
59	Parser	ParserImpl.mesa
60	Parser	ParserImpl.mesa
61	Parser	ParserImpl.mesa
62	Parser	ParserImpl.mesa
63	Parser	ParserImpl.mesa
64	Parser	ParserImpl.mesa
65	Display	StructChartImpl.mesa
66	MStream	StructChartImpl.mesa
67	Stream	StructChartImpl.mesa
68	Stream	StructChartImpl.mesa
69	Stream	StructChartImpl.mesa
70	String	StructChartImpl.mesa
71	String	StructChartImpl.mesa
72	String	StructChartImpl.mesa
		PROCEDURE CALLED

		ScanReset
		StringEqual
		ProcessQueue
		BaseTable
		Equal
		Pass2
		Pass3
		Pass4
		Pass5
		StateSeq
		LinkSeq
		ValueSeq
		AssignDescriptors
		ActionSeq
		DataFlow
		SetLength
		GetPosition
		PutChar
		PutString
		Length
		MakeString
		AppendSubString

STRUCTURE CHART - TABULAR OUTPUT LISTING

CLIENT

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
SLC		GenTextDisplay	FormSW	DesignExtractor
SLC		ClearOutputFile	FormSW	DesignExtractor
SLC		Run	FormSW	DesignExtractor
SLC		EnumerateFiles	FormSW	DesignExtractor
SLC		FreeNameDescriptor	FormSW	DesignExtractor
SLC	G	GetTableBase	Runtime	DesignExtractor
SLC	G	AllocateAtomStack	-	DesignExtractor
SLC		ExternalStoreList	-	DesignExtractor
SLC		Initialize	-	DesignExtractor
2	LG	Enter	-	DesignExtractor
2	LG	Exit	-	DesignExtractor
2	G	Delete	Heap	DesignExtractor
2	LG	Create	Heap	DesignExtractor
2	LG	Detach	Process	DesignExtractor
2		TextDisplay	-	DesignExtractor

SUBORDINATES

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
2	LG	Enter	-	DesignExtractor
2	LG	Exit	-	DesignExtractor
2	G	Delete	Heap	DesignExtractor
2	LG	Create	Heap	DesignExtractor
2	LG	Detach	Process	DesignExtractor
2		TextDisplay	-	DesignExtractor
2	LG	Enter	-	DesignExtractor
2	LG	Exit	-	DesignExtractor
2	LG	Detach	Process	DesignExtractor
2		ClearIt	-	DesignExtractor
2	LG	Enter	-	DesignExtractor
2	LG	Exit	-	DesignExtractor
2	LG	Detach	Process	DesignExtractor
2		RunIt	-	DesignExtractor
2		Init	ChangeTable	DesignExtractor
2		MakeTable	ChangeTable	DesignExtractor
2	G	Add	fileList	DesignExtractor
2	G	EnumeratedDirectory	MFile	DesignExtractor
2		AddEnumerating	-	DesignExtractor
2		AllEntryNames	fileList	DesignExtractor
2	G	CleanUp	ChangeTable	DesignExtractor
2	LG	Create	MSegment	DesignExtractor
2	TLG	Address	MSegment	DesignExtractor
2	TLG	CopyToNewString	String	DesignExtractor
2		AppendMessage	HeraldWindow	DesignExtractor
2	TLG	Create	Heap	DesignExtractor
2		AddCommand	Exec	DesignExtractor
2		ExecutiveProc	-	DesignExtractor
2		Unload	-	DesignExtractor
2		Copy	String	DesignExtractor
2	G	FreeString	String	DesignExtractor
2	LG	CopyToNewString	String	DesignExtractor
2		MakeSWs	-	DesignExtractor
2	G	log	-	DesignExtractor
3	TLG	Delete	Heap	DesignExtractor

STRUCTURE CHART - TABULAR OUTPUT LISTING

CLIENT

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
2		ClearIt	-	DesignExtractor
2		RunIt	-	DesignExtractor
2		Init	ChangeTable	DesignExtractor
2		MakeTable	ChangeTable	DesignExtractor
2	G	Add	fileList	DesignExtractor
2	G	EnumeratedDirectory	MFile	DesignExtractor
2		AddEnumerating	-	DesignExtractor
2		AllEntryNames	fileList	DesignExtractor
2	G	CleanUp	ChangeTable	DesignExtractor
2	TLG	Address	MSegment	DesignExtractor
2	TLG	CopyToNewString	String	DesignExtractor
2		AppendMessage	HeraldWindow	DesignExtractor

SUBORDINATES

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
3	TLG	Exit	-	DesignExtractor
3	LG	SetPriority	Process	DesignExtractor
3	LG	Yield	Process	DesignExtractor
3	TLG	log	-	DesignExtractor
3		CheckSelection	-	DesignExtractor
3	LG	CheckForAbort	-	DesignExtractor
3		Control	Display	DesignExtractor
3	TLG	Exit	-	DesignExtractor
3	LG	SetPriority	Process	DesignExtractor
3	LG	Yield	Process	DesignExtractor
3	TLG	log	-	DesignExtractor
3	LG	FreeAtomStack	-	DesignExtractor
3	G	AllocateAtomStack	-	DesignExtractor
3	LG	Delete	Heap	DesignExtractor
3	LG	CheckForAbort	-	DesignExtractor
3		CleanMetFile	Utils	DesignExtractor
3	TLG	Exit	-	DesignExtractor
3	LG	SetPriority	Process	DesignExtractor
3	LG	Yield	Process	DesignExtractor
3	TLG	log	-	DesignExtractor
3	G	Create	Heap	DesignExtractor
3	TLG	Delete	Heap	DesignExtractor
3		OpenMetFile	Utils	DesignExtractor
3		MetFileLength	Utils	DesignExtractor
3	TLG	CloseMetFile	Utils	DesignExtractor
3	TLG	FreeStringHandle	token	DesignExtractor
3		MaybeQuoted	token	DesignExtractor
3		Empty	String	DesignExtractor
3	TLG	FreeTokenString	Token	DesignExtractor
3		AppendExtensionIfNee	-	DesignExtractor
3	G	EnumeratedDirectory	MFile	DesignExtractor
3		ProcessFile	-	DesignExtractor
3	TLG	LongNumber	Format	DesignExtractor
3		ForceOutput	TextSw	DesignExtractor
3	G	Add	fileList	DesignExtractor
3	G	delete	file	DesignExtractor
3	TLG	log	-	DesignExtractor
3	G	CardToSString	Utils	DesignExtractor

STRUCTURE CHART - TABULAR OUTPUT LISTING

CLIENT

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
2		AddCommand	Exec	DesignExtractor
2		ExecutiveProc	-	DesignExtractor
2		Unload	-	DesignExtractor
2		Copy	String	DesignExtractor
2	G	FreeString	String	DesignExtractor
2		MakeSws	-	DesignExtractor
2	G	log	-	DesignExtractor
3	TLG	Delete	Heap	DesignExtractor
3	TLG	Exit	-	DesignExtractor
3	LG	SetPriority	Process	DesignExtractor
3	LG	Yield	Process	DesignExtractor
3	TLG	log	-	DesignExtractor
3		CheckSelection	-	DesignExtractor
3	LG	CheckForAbort	-	DesignExtractor
3		Control	Display	DesignExtractor
3	LG	FreeAtomStack	-	DesignExtractor
3	G	AllocateAtomStack	-	DesignExtractor
3		ClearMetFile	Utils	DesignExtractor
3	G	Create	Heap	DesignExtractor
3		OpenMetFile	Utils	DesignExtractor
3		MetFileLength	Utils	DesignExtractor
3	TLG	CloseMetFile	Utils	DesignExtractor
3	TLG	FreeStringHandle	token	DesignExtractor
3		MaybeQuoted	token	DesignExtractor
3		Empty	String	DesignExtractor
3	TLG	FreeTokenString	Token	DesignExtractor

SUBORDINATES

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
3		Activate	ToolWindow	DesignExtractor
3	G	Enter	-	DesignExtractor
3	LG	Text	Format	DesignExtractor
3		OutputProc	Exec	DesignExtractor
3	TLG	Exit	-	DesignExtractor
3		Destroy	Tool	DesignExtractor
3		RemoveCommand	Exec	DesignExtractor
3	TLG	Delete	Heap	DesignExtractor
3	LG	FreeAtomStack	-	DesignExtractor
3		MakeFormSW	Tool	DesignExtractor
3		MakeForm	-	DesignExtractor
3		UnusedLogName	Tool	DesignExtractor
3		MakeFileSW	Tool	DesignExtractor
3		NoteSws	ToolDriver	DesignExtractor
3	LG	Text	Put	DesignExtractor
4	LG	Text	Put	DesignExtractor
4	TLG	log	-	DesignExtractor
4	LG	UserAbort	UserInput	DesignExtractor
4		RawData	Display	DisplayImpl
4		StructChart	Display	DisplayImpl
4	LG	Address	MSegment	DesignExtractor
4	LG	Delete	MSegment	DesignExtractor
4	LG	Create	MSegment	DesignExtractor
4	TLG	Address	MSegment	DesignExtractor

STRUCTURE CHART - TABULAR OUTPUT LISTING

CLIENT

SUBORDINATES

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME	LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
3		AppendExtensionIfNee	-	DesignExtractor	4		EquivalentSubStrings	String	DesignExtractor
3	G	EnumeratedDirectory	MFile	DesignExtractor	4		to	-	DesignExtractor
3		ProcessFile	-	DesignExtractor	4	G	AppendStringAndGrow	String	DesignExtractor
3	TLG	LongNumber			4	TLG	log	-	DesignExtractor
3		ForceOutput			4	TLG	CleanUp	-	DesignExtractor
3	G	Add			4	G	CheckForAbort	-	DesignExtractor
3	G	delete			4	TLG	Create	MStream	DesignExtractor
3	G	CardIoString			4		Line	Format	DesignExtractor
3		Activate			4	TLG	CopyToNewString	String	DesignExtractor
3	G	Enter			4	G	CardIoString	Utils	DesignExtractor
3	LG	Text			4	G	FreeString	String	DesignExtractor
3		OutputProc			4		SetPosition	Stream	DesignExtractor
3		Destroy			4	TLG	Delete	Heap	DesignExtractor
3		RemoveCommand			4		ScanInit	Parser	DesignExtractor
3		MakeFormSW			4		Parse	Parser	DesignExtractor
3		MakeForm							
3				DesignExtractor	4		AllocateItemDescript	FormSW	DesignExtractor
3		Format		DesignExtractor	4		StringItem	FormSW	DesignExtractor
3		TextSW		DesignExtractor	4	TLG	BooleanItem	FormSW	DesignExtractor
3		fileList		DesignExtractor	4	TLG	CommandItem	FormSW	DesignExtractor
3		file		DesignExtractor					
3		Utils		DesignExtractor					
3		ToolWindow		DesignExtractor					
3		-		DesignExtractor					
3		Format		DesignExtractor					
3		Exec		DesignExtractor					
3		Tool		DesignExtractor					
3		Exec		DesignExtractor					
3		Tool		DesignExtractor					
3		-		DesignExtractor					
3		UnusedLogName		DesignExtractor					
3		MakeFileSW		DesignExtractor					
3		NoteSWs		DesignExtractor					
4	LG	Text		DesignExtractor					
4	TLG	log		DesignExtractor					
4	LG	UserAbort		DesignExtractor	5	LG	Text	Put	DesignExtractor
4		RawData		DisplayImpl					
4		StructChart		DisplayImpl					
5	G	CheckForAbort		Utils					StructChartImpl
5		WriteOnly		MStream					StructChartImpl
5	TLG	delete		statsFile					StructChartImpl
5		FormatData		-					StructChartImpl
5	TLG	log		Utils					StructChartImpl
5		DataFlow		Display					StructChartImpl

STRUCTURE CHART - TABULAR OUTPUT LISTING

=====

CLIENT

				SUBORDINATES					
LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME	LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
4	LG	Address	MSegment	DesignExtractor	5		SetLength	MStream	StructChartImpl
4	LG	Delete	MSegment	DesignExtractor	5		GetPosition	Stream	StructChartImpl
4	LG	Create	MSegment	DesignExtractor					
4		EquivalentSubStrings	String	DesignExtractor					
4		to	-	DesignExtractor					
4	G	AppendStringAndGrow	String	DesignExtractor					
4	TLG	CleanUp	-	DesignExtractor					
4	G	CheckForAbort	-	DesignExtractor	5	LG	delete	file	DesignExtractor
4		Line	Format	DesignExtractor	5	TLG	log	-	DesignExtractor
4	TLG	CopyToNewString	String	DesignExtractor	5	LG	CardToString	Utils	DesignExtractor
4	G	CardToString	Utils	DesignExtractor					
4	G	FreeString	String	DesignExtractor					
4		SetPosition	Stream	DesignExtractor					
4		ScanInit	Parser	DesignExtractor					
4		Parse	Parser	DesignExtractor					
					5		ParseInit	-	ParserImpl
					5		ParseReset	-	ParserImpl
					5	TLG	CopyToNewString	String	ParserImpl
					5	TLG	log	Utils	ParserImpl
					5		StringEqual	Utils	ParserImpl
					5		ProcessQueue	Parser	ParserImpl
					5		BaseTable	Parser	ParserImpl
					5	G	Equal	String	ParserImpl
					5	TLG	ExpandStack	-	ParserImpl
					5	TLG	Input	-	ParserImpl
					5	G	ExpandQueue	-	ParserImpl
					5		Number	Format	ParserImpl
					5		Pass2	Parser	ParserImpl
					5		Pass3	Parser	ParserImpl
					5		Pass4	Parser	ParserImpl
					5		Pass5	Parser	ParserImpl
					6	G	UserAbort	UserInput	DesignExtractor
					6	TLG	WriteFiles	-	StructChartImpl
					6		log	Utils	StructChartImpl
					6		WriteExportInfo	-	StructChartImpl

STRUCTURE CHART - TABULAR OUTPUT LISTING

CLIENT

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
5	TLG	log	Utils	StructChartImpl
5		DataFlow	Display	StructChartImpl
5		SetLength	MStream	StructChartImpl
5		GetPosition	Stream	StructChartImpl
5	LG	CardIoString	Utils	DesignExtractor
5	G	UserAbort	UserInput	DesignExtractor
5		ParseInit	-	ParserImpl
5	TLG	ParserReset	-	ParserImpl
5	TLG	CopyToNewString	String	ParserImpl
5		StringEqual	Utils	ParserImpl
5		ProcessQueue	Parser	ParserImpl
5		BaseTable	Parser	ParserImpl
5	G	Equal	String	ParserImpl
5	TLG	ExpandStack	-	ParserImpl
5	TLG	input	-	ParserImpl
5	G	ExpandQueue	-	ParserImpl
5		Number	Format	ParserImpl
5		Pass2	Parser	ParserImpl
5		Pass3	Parser	ParserImpl
5		Pass4	Parser	ParserImpl
5		Pass5	Parser	ParserImpl
6	G	UserAbort	UserInput	DesignExtractor
6		WriteFiles	-	StructChartImpl
6	TLG	log	Utils	StructChartImpl

SUBORDINATES

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
6	G	CheckForAbort	Utils	StructChartImpl
6		CheckReturnCondition	-	StructChartImpl
6		CheckIfTerminalDecl	-	StructChartImpl
6	TLG	CleanUpPositionData	-	StructChartImpl
6	TLG	CollectPositionData	-	StructChartImpl
6	TLG	UpdateFullStack	-	StructChartImpl
6	TLG	SuppressOutput	-	StructChartImpl
6		SearchForDeclaration	-	StructChartImpl
6		GoFindCallParams	-	StructChartImpl
6		StudyRepeats	-	StructChartImpl
6	G	WritesCTitles	-	StructChartImpl
6		SemiGraphOutput	-	StructChartImpl
6		TabularOutput	-	StructChartImpl
6	LG	Text	Put	DesignExtractor
6	G	GetTableBase	Runtime	ParserImpl
6	G	ExpandStack	-	ParserImpl
6	G	ExpandQueue	-	ParserImpl
6	LG	EraseQueue	-	ParserImpl
6	LG	EraseStack	-	ParserImpl
6	LG	AssignDescriptors	Parser	ParserImpl
6	LG	EraseQueue	-	ParserImpl
6	LG	AssignDescriptors	Parser	ParserImpl
7	TLG	StatsWrite	-	StructChartImpl
7	TLG	PutChar	Stream	StructChartImpl
7	TLG	CheckLength	-	StructChartImpl
7	TLG	FreeString	String	StructChartImpl

STRUCTURE CHART - TABULAR OUTPUT LISTING

===== CLIENT =====

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
6		WriteExportInfo	-	StructChartImpl
6	G	CheckForAbort	Utils	StructChartImpl
6		CheckReturnCondition	-	StructChartImpl
6		CheckIfTerminalDecl	-	StructChartImpl
6	TLG	CleanUpPositionData	-	StructChartImpl
6	TLG	CollectPositionData	-	StructChartImpl
6	TLG	UpdateFullStack	-	StructChartImpl
6	TLG	SuppressOutput	-	StructChartImpl
6		SearchForDeclaration	-	StructChartImpl
6		GoFindCallParams	-	StructChartImpl
6		StudyRepeats	-	StructChartImpl
6	G	WritesCTitles	-	StructChartImpl
6		SemiGraphOutput	-	StructChartImpl
6		TabularOutput	-	StructChartImpl

SUBORDINATES

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
7	LG	Text	Put	DesignExtractor
7	LG	CheckProcExists	-	StructChartImpl
7	LG	FreeString	String	StructChartImpl
7		OutputExportInfo	-	StructChartImpl
7	G	UserAbort	UserInput	DesignExtractor
7	TLG	StatsWrite	-	StructChartImpl
7	TLG	CopyToNewString	String	StructChartImpl
7	TLG	CardToString	Utils	StructChartImpl
7	TLG	CheckLength	-	StructChartImpl
7	TLG	FreeString	String	StructChartImpl
7	TLG	PutChar	Stream	StructChartImpl
7	TLG	StatsWrite	-	StructChartImpl
7	TLG	FreeString	String	StructChartImpl
7	LG	PutChar	Stream	StructChartImpl
7	LG	CardToString	Utils	StructChartImpl
7	TLG	log	Utils	StructChartImpl
7	LG	FindFileNumber	-	StructChartImpl
7	TLG	StatsWrite	-	StructChartImpl
7	TLG	CopyToNewString	String	StructChartImpl
7	TLG	CardToString	Utils	StructChartImpl
7	TLG	CheckLength	-	StructChartImpl
7	TLG	FreeString	String	StructChartImpl
7	LG	PutChar	Stream	StructChartImpl
7	LG	CheckProcExists	-	StructChartImpl
7	LG	FreeString	String	StructChartImpl
7	LG	FindFileNumber	-	StructChartImpl
7	TLG	PutChar	Stream	StructChartImpl
7	TLG	StatsWrite	-	StructChartImpl
7	TLG	CopyToNewString	String	StructChartImpl
7	TLG	CardToString	Utils	StructChartImpl
7	TLG	CheckLength	-	StructChartImpl
7	TLG	FreeString	String	StructChartImpl
7	LG	PutChar	Stream	StructChartImpl
7	LG	WritesCTitles	-	StructChartImpl

STRUCTURE CHART - TABULAR OUTPUT LISTING

CLIENT

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
6	LG	Text	Put	DesignExtractor
6	G	GetTableBase	Runtime	ParserImpl
6	G	ExpandStack	-	ParserImpl
6	G	ExpandQueue	-	ParserImpl
6	LG	EraseQueue	-	ParserImpl
6	LG	EraseStack	-	ParserImpl
6	LG	AssignDescriptors	Parser	ParserImpl
7	TLG	StatsWrite	-	StructChartImpl
7	TLG	PutChar	Stream	StructChartImpl
7	TLG	CheckLength	-	StructChartImpl
7	TLG	FreeString	String	StructChartImpl
7	LG	Text	Put	DesignExtractor
7	LG	CheckProcExists	-	StructChartImpl
7	LG	OutputExportInfo	-	StructChartImpl
7	G	UserAbort	UserInput	DesignExtractor
7	TLG	CopyToNewString	String	StructChartImpl
7	TLG	CardToCString	Utils	StructChartImpl
7	TLG	log	Utils	StructChartImpl
7	LG	FindFileNumber	-	StructChartImpl
7	G	WriteSCTitles	-	StructChartImpl
7	G	WriteTableTitles	-	StructChartImpl
7	TLG	OutputTableLine	-	StructChartImpl

SUBORDINATES

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
7	G	WriteTableTitles	-	StructChartImpl
7	TLG	OutputTableLine	-	StructChartImpl
7	LG	PutChar	Stream	StructChartImpl
7	G	EraseStack	-	ParserImpl
7	LG	AssignDescriptors	Parser	ParserImpl
7	G	EraseQueue	-	ParserImpl
7	LG	AssignDescriptors	Parser	ParserImpl
8	LG	PutString	Stream	StructChartImpl
8	LG	Length	String	StructChartImpl
8	LG	AddSpaces	-	StructChartImpl
8	LG	ReduceLength	-	StructChartImpl
8	TLG	WriteExportTitles	-	StructChartImpl
8	LG	CopyToNewString	String	StructChartImpl
8	LG	CardToCString	Utils	StructChartImpl
8	TLG	CheckLength	-	StructChartImpl
8	TLG	StatsWrite	-	StructChartImpl
8	TLG	FreeString	String	StructChartImpl
8	G	Equal	String	StructChartImpl
8	TLG	PutChar	Stream	StructChartImpl
8	LG	Text	Put	DesignExtractor
8	TLG	PutChar	Stream	StructChartImpl
8	TLG	StatsWrite	-	StructChartImpl
8	TLG	PutChar	Stream	StructChartImpl
8	TLG	StatsWrite	-	StructChartImpl
8	TLG	StatsWrite	-	StructChartImpl
8	TLG	CopyToNewString	String	StructChartImpl
8	TLG	CardToCString	Utils	StructChartImpl
8	TLG	CheckLength	-	StructChartImpl

STRUCTURE CHART - TABULAR OUTPUT LISTING

=====

CLIENT

SUBORDINATES

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME	LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
7	G	EraseStack	-	ParserImpl	8	TLG	FreeString	String	StructChartImpl
7	LG	AssignDescriptors	Parser	ParserImpl	8	TLG	WriteRepeatInfo	-	StructChartImpl
7	G	EraseQueue	-	ParserImpl	8	TLG	FindFileNumber	-	StructChartImpl
8	LG	PutString	Stream	StructChartImpl	8	TLG	Length	String	StructChartImpl
8	LG	Length	String	StructChartImpl	8	TLG	ReduceLength	-	StructChartImpl
8	LG	AddSpaces	-	StructChartImpl	8	TLG	PutChar	Stream	StructChartImpl
8	LG	ReduceLength	-	StructChartImpl	8	TLG	WriteTableTitles	-	StructChartImpl
8	TLG	WriteExportTitles	-	StructChartImpl	9	LG	AppendStringAndGrow	String	StructChartImpl
8	LG	CopyToNewString	String	StructChartImpl	9	LG	MakeString	String	StructChartImpl
8	LG	CardToString	Utils	StructChartImpl	9	LG	AppendSubString	String	StructChartImpl
8	TLG	CheckLength	-	StructChartImpl	9	TLG	FreeString	String	StructChartImpl
8	TLG	StatsWrite	-	StructChartImpl	9	LG	CopyToNewString	String	StructChartImpl
8	TLG	FreeString	String	StructChartImpl	9	TLG	StatsWrite	-	StructChartImpl
8	G	Equal	String	StructChartImpl	9	LG	Length	String	StructChartImpl
8	TLG	PutChar	Stream	StructChartImpl	9	LG	AddSpaces	-	StructChartImpl
8	LG	Text	Put	DesignExtractor	9	LG	ReduceLength	-	StructChartImpl
8	TLG	WriteRepeatInfo	-	StructChartImpl	9	LG	PutString	Stream	StructChartImpl
8	TLG	FindFileNumber	-	StructChartImpl	9	LG	CopyToNewString	String	StructChartImpl
8	TLG	WriteTableTitles	-	StructChartImpl	9	TLG	AppendStringAndGrow	String	StructChartImpl
9	LG	AppendStringAndGrow	String	StructChartImpl	9	LG	CheckLength	-	StructChartImpl
9	LG	MakeString	String	StructChartImpl	9	TLG	StatsWrite	-	StructChartImpl
9	LG	AppendSubString	String	StructChartImpl	9	LG	FreeString	String	StructChartImpl
9	TLG	FreeString	String	StructChartImpl	9	TLG	PutChar	Stream	StructChartImpl
9	LG	CopyToNewString	String	StructChartImpl	9	TLG	StatsWrite	-	StructChartImpl
9	TLG	StatsWrite	-	StructChartImpl	9	TLG	PutChar	Stream	StructChartImpl

STRUCTURE CHART - TABULAR OUTPUT LISTING

=====

CLIENT

SUBORDINATES

LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME	LVL	REP	PROCEDURE NAME	INTERFACE NAME	FILE NAME
9	TLG	PutChar	Stream	StructChartImpl	10	LG	PutString	Stream	StructChartImpl
9	LG	Length	String	StructChartImpl					
9	LG	AddSpaces	-	StructChartImpl					
9	LG	ReduceLength	-	StructChartImpl	10	LG	AppendStringAndGrow	String	StructChartImpl
9	LG	PutString	Stream	StructChartImpl					
9	LG	CheckLength	-	StructChartImpl	10	LG	Length	String	StructChartImpl
					10	LG	AddSpaces	-	StructChartImpl
					10	LG	ReduceLength	-	StructChartImpl
10	LG	PutString	Stream	StructChartImpl					
10	LG	AppendStringAndGrow	String	StructChartImpl					
10	LG	MakeString	String	StructChartImpl					
10	LG	AppendSubString	String	StructChartImpl					
10	TLG	FreeString	String	StructChartImpl					
10	LG	CopyToNewString	String	StructChartImpl					
10	LG	Length	String	StructChartImpl					
10	LG	AddSpaces	-	StructChartImpl					
10	LG	ReduceLength	-	StructChartImpl	11	LG	AppendStringAndGrow	String	StructChartImpl
11	LG	AppendStringAndGrow	String	StructChartImpl					
11	LG	MakeString	String	StructChartImpl	11	LG	MakeString	String	StructChartImpl
11	LG	AppendSubString	String	StructChartImpl	11	LG	AppendSubString	String	StructChartImpl
11	TLG	FreeString	String	StructChartImpl	11	TLG	FreeString	String	StructChartImpl
11	LG	CopyToNewString	String	StructChartImpl	11	LG	CopyToNewString	String	StructChartImpl

APPENDIX A - OUTPUT LISTINGS

A.3

OUTPUT LISTING OF DATAFLOW.DATA

A.3.1 DFD VARIABLE

Mesa Source Files: DesignExtractorImpl
ParserImpl DisplayImpl StructChartImpl

Table 35

Repeats

Updating DataBase with DataFlow Info

```

~ ~ ~ Display Complete ~ ~ ~

```

Processing Display

Creating Structure Chart.....

- Studying repeat calls..
- Creating Semi Graphical Output.

Creating DataFlow Chart.

- Creating variable table,....

- Constructing Context Level diagram.

- Constructing level zero diagram....

- Constructing Lower Level diagram...

- writing dataflow table,...

Done Output created to file DataFlow.data

[illegible]

~ ~ ~ Display Complete ~ ~ ~

DESIGN EXTRACTION STATISTICS

=====

The files parsed in this session were:

Index	File Name	Start Index	End Index	Exports to
1	DesignExtractorImpl.mesa	1	3778	Utils
2	DisplayImpl.mesa	3779	3945	Display
3	ParserImpl.mesa	3946	5725	Parser
4	StructChartImpl.mesa	5726	12305	Display

VARIABLE OUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
1	DesignExtractorImp1										
3			0	M	0	1	384	0	ExternalProps	DataBase	
4			0	M	0	1	480	478	systemZone	DataBase	
11			0	M	0	1	684	0	Other	DataBase	
248			0	G	1	0	3708	657	tablePtr	DataBase	
249			0	M	0	1	3716	0	MesaTab	Call Param	GetTableBase
250			0	G	1	0	3720	432	fileStackPtr	DataBase	
251			0	M	0	1	3728	3726	FileStack	DataBase	
252			0	G	1	1	3731	464	atomIndexPtr	DataBase	
253			0	M	0	1	3739	3737	AtomIndex	DataBase	
254			0	G	1	0	3742	444	extStackPtr	DataBase	
255			0	G	0	1	3748	373	ExternalStack	DataBase	
256			0	G	0	1	3752	466	sh	DataBase	
257			0	G	0	1	3754	448	atomStackPtr	DataBase	
258			0	M	0	1	3761	3495	MaxString	Call Param	AllocateAtomStack
5		Enter					482	0			
6			1	R	0	1	501	488	ok	Return Valu	
7			1	G	0	1	504	352	running	Return Valu	
8			1	G	1	0	509	352	running	Return Valu	
9		Exit					517	0			
10			2	G	1	0	528	352	running	DataBase	
12		log					706	706			
13			3	P	0	1	718	548	fileSW	Call Param	Text
14			3	M	0	1	720	0	s	Call Param	Text
15		ExecutiveProc					725	0			
16			4	P	0	1	736	544	window	Call Param	Activate
17		Unload					741	0			
18			5	M	0	1	763	0	h	Call Param	OutputProc
19			5	M	0	1	771	0	abort	Return Valu	
20			5	P	0	1	786	544	window	DataBase	
21			5	F	1	1	796	544	window	Call Param	Destroy
22			5	M	0	1	809	0	h	Call Param	RemoveCommand
23			5	G	0	1	811	665	execCommand	Call Param	RemoveCommand
24			5	P	0	1	816	536	tempZone	DataBase	
25			5	F	1	1	826	536	tempZone	Call Param	Delete
26			5	G	0	1	837	534	permZone	Call Param	Delete
27			5	G	1	1	847	534	permZone	DataBase	
28			5	G	0	1	858	448	atomStackPtr	Call Param	Delete
29			5	G	0	1	866	3622	atomStackPtr	Call Param	FreeAtomStack
30			5	G	0	1	868	3628	sh	Call Param	FreeAtomStack
31			5	G	0	2	875	432	fileStackPtr	DataBase	
32			5	G	0	2	893	454	atomIndexPtr	DataBase	
33		CheckForAbort					917	706			
34			6	P	0	1	928	544	window	Call Param	UserAbort
35		ProcessFile					936	0			
54			7	M	0	4	1126	1124	Error	DataBase	
55			7	M	0	1	1134	1132	BoundsFault	DataBase	
56			7	G	0	3	1143	692	error	DataBase	
57			7	M	0	1	1181	0	release	DataBase	
58			7	M	0	1	1204	0	log	Call Param	Line
59			7	M	0	1	1206	0	name	Call Param	Line

VARIABLE OUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
60			7	M	0	1	1221	0	name	Call Param	log
61			7	G	0	1	1240	480	MaxFiles	DataBase	
62			7	L	1	0	1248	966	tempString	DataBase	
63			7	G	0	1	1260	460	MaxFiles	Call Param	CardToString
64			7	P	0	1	1263	472	zone	DataBase	
65			7	L	0	1	1268	966	tempString	Call Param	log
66			7	P	0	2	1275	472	zone	Call Param	FreeString
67			7	L	0	1	1277	966	tempString	Call Param	FreeString
68			7	D	1	0	1292	1287	fileCount	DataBase	
69			7	M	0	1	1298	1296	FreeString	DataBase	
70			7	D	0	1	1307	1302	fileName	DataBase	
71			7	D	1	0	1315	1310	fileName	DataBase	
72			7	M	0	1	1321	0	name	Call Param	CopyToNewString
73			7	P	0	1	1323	472	zone	Call Param	CopyToNewString
74			7	D	1	0	1331	1326	startIndex	DataBase	
75			7	D	0	1	1340	1333	endIndex	DataBase	
76			7	L	1	1	1357	959	parseZone	DataBase	
77			7	L	1	1	1394	959	parseZone	Call Param	Delete
78			7	G	0	1	1406	657	tablePtr	Call Param	ScanInit
79			7	L	0	1	1408	1167	file	Call Param	ScanInit
80			7	L	0	2	1410	959	parseZone	Call Param	Parse
81			7	L	0	1	1421	4520	parseZone	Call Param	Parse
82			7	G	0	1	1423	4525	atomStackPtr	Call Param	Parse
83			7	G	0	1	1425	4531	atomIndexPtr	Call Param	Parse
84			7	G	0	1	1427	4537	fileStackPtr	Call Param	Parse
85			7	G	0	1	1429	4543	fileCount	Call Param	Parse
86			7	P	0	1	1431	4547	rowData	Call Param	Parse
87			7	D	1	0	1439	1434	endIndex	DataBase	
88			7	L	1	1	1458	959	parseZone	Call Param	Delete
36		CleanUp					971	0	file	DataBase	
37			8	E	0	1	990	943	delete	DataBase	
38			8	M	0	1	998	996	delete	DataBase	
39			8	E	2	1	1006	943	file	Call Param	delete
40			8	I	0	1	1020	975	ok	Call Param	log
41			8	L	1	3	1024	951	errors	Call Param	log
42			8	L	0	2	1030	953	warnings	Call Param	log
43			8	M	0	2	1046	0	log	DataBase	
44			8	M	0	2	1050	1048	CardToString	DataBase	
45			8	G	0	7	1052	432	fileStackPtr	DataBase	
46			8	G	0	1	1054	438	fileCount	DataBase	
47			8	D	0	1	1057	1052	endIndex	DataBase	
48			8	G	1	10	1072	438	fileCount	Call Param	CardToString
49			8	G	0	2	1092	448	atomStackPtr	DataBase	
50			8	G	0	2	1094	454	atomIndexPtr	DataBase	
51			8	D	0	1	1098	1092	sCount	DataBase	
52			8	G	1	1	1107	641	totalErrors	DataBase	
53			8	G	1	1	1113	643	totalWarnings	DataBase	
89		GenTextDisplay					1483	0			
90			9	P	0	1	1504	536	tempZone	DataBase	
91			9	P	2	1	1513	536	tempZone	Call Param	Delete
92		TextDisplay					1561	0			
93			10	F	0	1	1571	536	tempZone	DataBase	

VARIABLE OUPUT TA8LE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
94			10	F	1	2	1580	538	tempZone	Call Param	Delete
95			10	P	0	1	1800	827	background	Database	
96			10	M	0	1	1809	1807	priority8background	Call Param	SetPriority
97			10	L	2	1	1830	1593	ok	Database	
98			10	G	0	1	1858	3798	created	Call Param	Control
99			10	G	0	1	1880	3802	atomStackPtr	Call Param	Control
100			10	G	0	1	1882	3808	atomIndexPtr	Call Param	Control
101			10	G	0	1	1884	3814	extStackPtr	Call Param	Control
102			10	G	0	1	1888	3819	maxExternal	Call Param	Control
103			10	G	0	1	1888	3823	fileStackPtr	Call Param	Control
104			10	G	0	1	1872	3829	MaxFiles	Call Param	Control
105			10	P	0	1	1874	3833	rawData	Call Param	Control
108			10	P	0	1	1878	3837	SCGraph	Call Param	Control
107			10	P	0	1	1878	3841	SCTable	Call Param	Control
108			10	P	0	1	1880	3845	DFDVariable	Call Param	Control
109			10	P	0	1	1882	3849	DFDTable	Call Param	Control
110			10	P	0	1	1884	3853	debugg	Call Param	Control
111			10	P	0	1	1888	3857	repeats	Call Param	Control
112			10	P	1	1	1704	538	tempZone	Call Param	Delete
113		CheckSelection					1724	0			
114			11	P	0	2	1748	585	rawData	Database	
115			11	P	0	2	1750	557	SCGraph	Database	
116			11	P	0	2	1752	584	SCTable	Database	
117			11	P	0	2	1754	571	DFDVariable	Database	
118			11	P	0	2	1758	578	DFDTable	Database	
119			11	L	1	0	1794	1733	ok	Database	
120			11	L	0	1	1828	1733	ok	Return Valu	
121		ClearOutputFile					1833	0			
122		ClearIt					1870	0			
123			13	P	0	1	1889	827	background	Database	
124			13	M	0	1	1898	1898	priority8background	Call Param	SetPriority
125			13	G	1	0	1919	438	fileCount	Database	
128			13	G	1	1	1928	432	fileStackPtr	Database	
127			13	G	0	1	1938	872	filename	Database	
128			13	G	0	1	1941	3822	atomStackPtr	Call Param	FreeAtomStack
129			13	G	0	1	1943	3828	sh	Call Param	FreeAtomStack
130			13	G	1	1	1951	454	atomIndexPtr	Database	
131			13	M	0	1	1982	1980	FileStack	Database	
132			13	M	0	1	1973	1971	AtomIndex	Database	
133			13	G	0	1	1977	488	sh	Database	
134			13	G	0	1	1979	448	atomStackPtr	Database	
135			13	M	0	1	1988	3495	MaxString	Call Param	AllocateAtomStack
136			13	P	0	1	1991	538	tempZone	Database	
137			13	P	1	1	2001	538	tempZone	Call Param	Delete
138			13	L	1	0	2015	1882	ok	Database	
139			13	L	0	1	2037	1882	ok	Call Param	log
140		Run					2050	0			
141		RunIt					2087	0			
142			15	P	0	1	2114	827	background	Database	
143			15	M	0	1	2123	2121	priority8background	Call Param	SetPriority
144			15	G	0	1	2141	850	toolName	Call Param	log
145			15	G	1	1	2149	841	totalErrors	Database	

VARIABLE OUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
146			15	G	1	1	2151	643	totalWarnings	DataBase	
147			15	P	1	1	2155	536	tempZone	DataBase	Delete
148			15	P	1	1	2192	536	tempZone	Call Param	OpenMetFile
149			15	M	0	1	2207	2205	Error	Call Param	OpenMetFile
150			15	M	0	1	2211	2209	Error	Call Param	
151			15	L	2	2	2219	2099	aborted	DataBase	
152			15	L	1	0	2225	2105	initialFileLength	DataBase	
153			15	G	0	1	2237	672	filename	DataBase	
154			15	L	2	1	2289	2271	token	DataBase	
155			15	M	0	1	2316	2314	UnterminatedQuote	Call Param	MaybeQuoted
156			15	L	1	2	2326	2301	s	Call Param	Empty
157			15	L	0	1	2346	2301	s	Call Param	FreeTokenString
158			15	L	1	0	2385	2357	s	DataBase	
159			15	L	0	1	2391	2357	s	Call Param	FreeTokenString
160			15	L	0	1	2431	2099	aborted	Call Param	log
161			15	M	0	1	2442	0	log	Call Param	LongNumber
162			15	G	0	1	2444	438	fileCount	Call Param	LongNumber
163			15	G	0	1	2453	438	fileCount	Call Param	log
164			15	M	0	1	2477	0	log	Call Param	LongNumber
165			15	G	0	1	2479	641	totalErrors	Call Param	LongNumber
166			15	G	0	1	2488	641	totalErrors	Call Param	log
167			15	M	0	1	2514	0	log	Call Param	LongNumber
168			15	G	0	1	2516	643	totalWarnings	Call Param	LongNumber
169			15	G	0	1	2525	643	totalWarnings	Call Param	log
170			15	P	1	1	2547	536	tempZone	Call Param	Delete
171			15	F	0	1	2558	548	fileSW	Call Param	ForceOutput
172		AppendExtensionIfNee	16	M	0	1	2594	2592	length	DataBase	
173			16	L	0	1	2668	2611	toSS	Call Param	EquivalentSubStrings
174			16	L	0	1	2671	2640	extSS	Call Param	EquivalentSubStrings
175			16	L	0	1	2679	2599	length	Call Param	to
176			16	L	0	1	2681	2601	minLength	Call Param	to
177		EnumerateFiles	17	L	1	0	2754	2722	fileList	DataBase	
181			17	M	1	0	2789	0	hints	DataBase	
182			17	M	0	1	2799	0	hints	Return Valu	
183			17	M	0	1	2801	0	FreeNameDescriptor	Return Valu	
184		AddEnumerating	18	L	0	1	2744	2370	name	Call Param	Add
179		FreeNameDescriptor	20	M	1	0	2822	0	freeDesc	DataBase	
180		MakeForm	20	M	1	10	2835	0	items	DataBase	
185			20	L	0	1	2845	2829	numItems	Call Param	AllocateItemDescript
186			20	M	0	1	2868	2866	line0	Call Param	StringItem
187			20	M	0	1	2871	0	string	DataBase	
188			20	M	0	1	2880	0	inHeap	DataBase	
189			20	M	0	1	2905	2903	sameLine	Call Param	BooleanItem
190			20	M	0	6	2908	0	switch	DataBase	
191			20	P	0	1	2911	557	SCGraph	DataBase	
192			20	M	0	6	2913	0	drawBox	DataBase	
193			20	M	0	1	2938	2936	sameLine	Call Param	BooleanItem
194			20	M	0	1	2938	2936	sameLine	Call Param	BooleanItem
195			20	M	0	1	2938	2936	sameLine	Call Param	BooleanItem
196			20	M	0	1	2938	2936	sameLine	Call Param	BooleanItem
197			20	M	0	1	2938	2936	sameLine	Call Param	BooleanItem

VARIABLE DUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
198			20	P	0	1	2944	571	DFDVariable	DataBase	
199			20	M	0	1	2971	2969	sameLine	Call Param	BooleanItem
200			20	P	0	1	2977	585	rawData	DataBase	
201			20	M	0	1	3004	3002	line1	Call Param	CommandItem
202			20	M	0	1	3032	3030	sameLine	Call Param	CommandItem
203			20	M	0	1	3060	3058	sameLine	Call Param	CommandItem
204			20	M	0	1	3088	3086	sameLine	Call Param	BooleanItem
205			20	P	0	1	3094	564	SCTable	DataBase	
206			20	M	0	1	3121	3119	sameLine	Call Param	BooleanItem
207			20	P	0	1	3127	578	DFDTable	DataBase	
208			20	M	0	1	3154	3152	sameLine	Call Param	BooleanItem
209			20	P	0	1	3160	606	repeats	DataBase	
210			20	M	0	1	3169	0	items	Return Valu	
211			20	M	0	1	3173	0	freeDesc	Return Valu	
212		MakeSWs	20	M	0	1	3180	0			
213			21	M	0	1	3214	3212	Address	DataBase	
214			21	P	1	0	3225	546	formSW	DataBase	
215			21	P	1	0	3253	548	fileSW	DataBase	
216			21	L	1	0	3268	3203	addresses	DataBase	
217			21	L	0	1	3304	3203	addresses	Call Param	NoteSWs
218		Initialize	22	M	0	1	3310	0			
219			22	M	0	1	3322	3320	LookupCommand	DataBase	
220			22	G	0	1	3324	665	execCommand	DataBase	
221			22	L	0	3	3327	3315	name	DataBase	
222			22	G	1	1	3361	534	permZone	DataBase	
223			22	P	0	1	3435	472	zone	Call Param	FreeString
224			22	G	1	1	3437	650	toolName	Call Param	FreeString
225			22	L	1	0	3457	3259	window	DataBase	
226			22	G	0	1	3456	650	toolName	Call Param	log
227		AllocateAtomStack	23	M	0	1	3491	0			
228			23	M	0	1	3533	3531	AtomStack	DataBase	
229			23	I	0	2	3539	3495	elements	DataBase	
230			23	M	0	1	3548	3546	AtomProps	DataBase	
231			23	M	0	2	3564	3562	wordsPerPage	DataBase	
232			23	R	1	0	3577	3501	sh	DataBase	
233			23	R	1	0	3593	3507	aSP	DataBase	
234			23	R	0	1	3599	3501	sh	Call Param	Address
235			23	L	1	1	3602	3515	length	DataBase	
236			23	R	0	1	3608	3501	sh	Call Param	Address
237		FreeAtomStack	24	I	1	1	3637	3622	aSP	DataBase	
238			24	I	0	1	3643	3628	sh	Call Param	Address
239			24	I	1	1	3656	3628	sh	Call Param	Delete
240		ExternalStoreList	25	G	1	2	3665	0			
241			25	G	1	1	3670	426	eSCount	DataBase	
242			25	G	0	2	3674	444	extStackPtr	DataBase	
243			25	D	1	0	3679	3674	extStoreType	DataBase	
244			25	P	0	1	3687	472	zone	Call Param	CopyToNewString
245			25	D	1	0	3695	3690	extStoreAction	DataBase	
246			25	P	0	1	3703	472	zone	Call Param	CopyToNewString
247		DisplayImpl	25	P	0	1	3793	6158			
248		Control	260								

VARIABLE OUTPUT TABLE

VSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
261			26	-	0	14	3798	0	created	Formal Para	
262			26	I	0	1	3874	3833	rawData	DataBase	
263			26	L	2	0	3878	3867	result	DataBase	
264			26	I	0	1	3882	3798	created	Call Param	RawData
265			26	I	0	1	3884	3802	aStackPtr	Call Param	RawData
266			26	I	0	1	3886	3808	fileStackPtr	Call Param	RawData
267			26	I	0	1	3888	3823	MaxFiles	Call Param	RawData
268			26	I	0	1	3890	3829	debugg	Call Param	RawData
269			26	I	0	1	3892	3853	SCGraph	DataBase	
270			26	I	0	1	3896	3837	SCTable	DataBase	
271			26	I	0	1	3898	3841	DFDVariable	DataBase	
272			26	I	0	1	3900	3845	DFDTable	DataBase	
273			26	I	0	1	3902	3849	created	Call Param	StructChart
274			26	I	0	1	3910	6163	aStackPtr	Call Param	StructChart
275			26	I	0	1	3912	6167	extStackPtr	Call Param	StructChart
276			26	I	0	1	3914	6173	maxExternal	Call Param	StructChart
277			26	I	0	1	3916	6179	fileStackPtr	Call Param	StructChart
278			26	I	0	1	3918	6184	MaxFiles	Call Param	StructChart
279			26	I	0	1	3920	6188	SCGraph	Call Param	StructChart
280			26	I	0	1	3922	6194	SCTable	Call Param	StructChart
281			26	I	0	1	3924	6198	DFDVariable	Call Param	StructChart
282			26	I	0	1	3926	6202	DFDTable	Call Param	StructChart
283			26	I	0	1	3928	6206	debugg	Call Param	StructChart
284			26	I	0	1	3930	6210	repeats	Call Param	StructChart
285			26	I	0	1	3932	6214	result	Return Valu	StructChart
286			26	I	0	1	3934	6218			
287			26	L	0	1	3939	3867			
288	ParserImp1										
436			0	M	0	1	5783	0	maxLevels	DataBase	
437			0	M	0	1	5786	0	PositionProps	DataBase	
438			0	M	0	1	5842	0	maxExportStack	DataBase	
439			0	M	0	1	5845	0	ExportStackProps	DataBase	
440			0	M	0	1	5866	5864	MaxString	DataBase	
441			0	M	0	1	6112	6110	systemZone	DataBase	
289		input					4105	0			
290		ParseInit					4321	0			
291			28	G	1	10	4328	4192	tablePtr	DataBase	
292			28	G	0	1	4334	3953	mesaTab	Call Param	GetTableBase
293			28	G	1	1	4338	4198	tStart	DataBase	
294			28	G	1	1	4350	4208	tLength	DataBase	
295			28	G	1	1	4362	4218	tSymbol	DataBase	
296			28	G	1	1	4374	4228	tAction	DataBase	
297			28	G	1	1	4386	4238	nStart	DataBase	
298			28	G	1	1	4398	4248	nLength	DataBase	
299			28	G	1	1	4410	4258	nSymbol	DataBase	
300			28	G	1	1	4422	4268	nAction	DataBase	
301			28	G	1	1	4434	4278	ntDefault	DataBase	
302			28	G	1	1	4446	4288	prodData	DataBase	
303			28	G	1	0	4458	4141	s	DataBase	
304			28	G	1	0	4462	4182	q	DataBase	
305		ParseReset					4478	0			
306		InputLoc					4499	0			

VARIABLE OUTPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
307			30	G	0	1	4511	4117	inputLoc	Return Valu	
308		Parse					4515	3793		Formal Para	
309			31	-	0	7	4520	0	cz	Database	
310			31	G	1	0	4586	4314	z	Database	
311			31	I	0	1	4588	4520	cz	Database	
312			31	M	1	0	4602	0	input	Database	
313			31	M	0	1	4606	4604	Atom	Database	
314			31	L	1	0	4608	4568	nErrors	Database	
315			31	L	1	0	4612	4564	complete	Database	
316			31	L	5	11	4616	4570	i	Database	
317			31	G	1	1	4618	4159	top	Database	
318			31	L	3	6	4620	4572	valid	Database	
319			31	G	1	0	4624	4188	qi	Database	
320			31	G	0	6	4628	4141	s	Database	
321			31	L	4	4	4633	4560	currentState	Database	
322			31	M	0	1	4635	0	InitialState	Database	
323			31	M	3	0	4639	4637	class	Database	
324			31	G	0	2	4641	4135	NullSymbol	Database	
325			31	G	1	2	4643	4101	inputSymbol	Database	
326			31	M	0	1	4645	0	InitialSymbol	Database	
327			31	G	1	0	4647	4121	inputValue	Database	CopyToNewString
328			31	G	0	1	4655	4314	z	Call Param	
329			31	G	1	3	4658	4117	inputLoc	Database	
330			31	P	0	1	4662	4163	ids	Database	
331			31	G	0	1	4673	4314	z	Call Param	CopyToNewString
332			31	P	1	0	4676	4175	idindex	Database	
333			31	M	0	1	4690	0	FinalState	Database	
334			31	M	0	1	4706	0	tokenId	Database	
335			31	G	0	1	4716	4121	inputValue	Call Param	StringEqual
336			31	P	0	2	4718	4163	ids	Call Param	StringEqual
337			31	P	2	4	4720	4175	idindex	Call Param	StringEqual
338			31	M	0	1	4732	4730	maxIds	Database	
339			31	G	0	1	4754	4121	inputValue	Call Param	CopyToNewString
340			31	G	0	1	4756	4314	z	Call Param	CopyToNewString
341			31	L	0	5	4762	4693	ti	Database	
342			31	G	0	1	4769	4208	tlength	Database	
343			31	G	0	1	4776	4218	tSymbol	Database	
344			31	M	0	1	4783	0	DefaultMarker	Database	
345			31	M	0	2	4793	0	SyntaxError	Database	
346			31	L	3	1	4797	4578	action	Database	
347			31	G	0	1	4799	4228	tAction	Database	
348			31	L	0	2	4832	4820	k	Database	
349			31	G	0	1	4848	4308	ruleValue	Database	
350			31	L	3	4	4855	4814	qi	Call Param	ProcessQueue
351			31	G	2	7	4857	4159	top	Call Param	ProcessQueue
352			31	I	0	1	4866	4525	aStackPtr	Database	
353			31	L	0	2	4868	4582	unique	Database	
354			31	G	0	1	4875	4121	inputValue	Call Param	BaseTable
355			31	G	0	4	4877	4101	inputSymbol	Call Param	BaseTable
356			31	I	0	6	4879	4531	aIndexPtr	Call Param	BaseTable
357			31	I	0	1	4881	4525	aStackPtr	Call Param	BaseTable
358			31	I	0	1	4883	4543	fileCount	Call Param	BaseTable

VARIABLE OUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
359			31	G	0	1	4885	4308	ruleValue	Call Param	BaseTable
360			31	G	0	4	4893	4121	inputValue	Call Param	Equal
361			31	M	0	4	4906	4904	MaxString	DataBase	
362			31	M	0	3	4936	4934	length	DataBase	
363			31	G	0	2	4949	4153	v	DataBase	
364			31	G	0	2	4956	4147	1	DataBase	
365			31	M	0	1	4982	4980	tag	DataBase	
366			31	G	0	1	4984	4091	Scan	DataBase	
367			31	G	0	1	4998	4182	q	DataBase	
368			31	M	0	1	5019	5015	pLength	DataBase	
369			31	D	0	1	6057	5023	lhs	DataBase	
370			31	M	0	1	5064	0	NTState	DataBase	
371			31	L	0	5	5078	5068	nI	DataBase	
372			31	G	0	1	5085	4248	nLength	DataBase	
373			31	L	0	2	5092	5046	lhs	DataBase	
374			31	G	0	1	5094	4258	nSymbol	DataBase	
375			31	G	0	1	5102	4268	nAction	DataBase	
376			31	M	0	2	5109	0	nFound	DataBase	
377			31	G	0	1	5117	4278	ntDefaults	DataBase	
378			31	L	1	1	5139	4574	m	DataBase	
379			31	M	0	1	5168	5166	transition	DataBase	
380			31	M	1	0	5176	5174	value	DataBase	
381			31	M	1	0	5185	5183	index	DataBase	
382			31	M	0	1	5248	5246	log	DataBase	
383			31	M	0	1	5252	5250	index	Call Param	Number
384			31	G	0	1	5294	4121	inputValue	Call Param	Number
385			31	G	0	1	5296	4101	inputSymbol	Call Param	Pass2
386			31	I	0	1	5298	4531	aIndexPtr	Call Param	Pass2
387			31	I	0	1	5300	4525	aStackPtr	Call Param	Pass2
388			31	I	0	1	5302	4537	fileStackPtr	Call Param	Pass2
389			31	I	0	1	5304	4543	fileCount	Call Param	Pass2
390			31	G	0	1	5306	4308	ruleValue	Call Param	Pass2
391			31	G	0	1	5323	4121	inputValue	Call Param	Pass3
392			31	G	0	1	5325	4101	inputSymbol	Call Param	Pass3
393			31	I	0	1	5327	4531	aIndexPtr	Call Param	Pass3
394			31	I	0	1	5329	4525	aStackPtr	Call Param	Pass3
395			31	I	0	1	5331	4537	fileStackPtr	Call Param	Pass3
396			31	I	0	1	5333	4543	fileCount	Call Param	Pass3
397			31	G	0	1	5335	4308	ruleValue	Call Param	Pass3
398			31	I	0	1	5349	4531	aIndexPtr	Call Param	Pass4
399			31	I	0	1	5351	4525	aStackPtr	Call Param	Pass4
400			31	I	0	1	5353	4537	fileStackPtr	Call Param	Pass4
401			31	I	0	1	5355	4543	fileCount	Call Param	Pass4
402			31	I	0	1	5369	4531	aIndexPtr	Call Param	Pass5
403			31	I	0	1	5371	4525	aStackPtr	Call Param	Pass5
404			31	I	0	1	5373	4537	fileStackPtr	Call Param	Pass5
405			31	I	0	1	5375	4543	fileCount	Call Param	Pass5
406			31	I	0	1	5386	4525	aStackPtr	Call Param	Pass5
407		ExpandStack					5390	0		Return Valu	
408			32	G	1	2	5405	4141	s	DataBase	
409			32	M	0	1	5413	5411	length	DataBase	
410			32	L	0	2	5419	5400	oldSize	DataBase	

VARIABLE OUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
411			32	I	0	1	5421	5394	delta	Database	
418			32	L	0	6	5490	5478	i	Database	
419			32	G	1	1	5503	4147	i	Database	
420			32	G	1	1	5513	4153	v	Database	
412		newS					5423	0			
413			33	L	0	1	5437	5415	newSize	Call Param	StatesSeq
414		newL					5441	0			
415			34	L	0	1	5455	5415	newSize	Call Param	LinkSeq
416		newV					5459	0			
417			35	L	0	1	5473	5415	newSize	Call Param	ValueSeq
421		EraseStack					5561	0			
422			36	G	0	2	5567	4141	s	Database	
423			36	G	0	1	5577	4153	v	Database	
424			36	G	0	1	5585	4147	i	Database	
425		ExpandQueue					5598	0			
426			37	G	1	2	5613	4182	q	Database	
427			37	M	0	1	5621	5619	length	Database	
428			37	L	0	2	5627	5608	oldSize	Database	
429			37	I	0	1	5629	5602	delta	Database	
432			37	L	0	2	5662	5650	i	Database	
430		newQ					5631	0			
431			38	L	0	1	5645	5623	newSize	Call Param	ActionSeq
433		EraseQueue					5705	0			
434			39	G	0	2	5711	4182	q	Database	
435	StructChartImp1										
437			0	M	0	1	5786	0	PositionProps	Database	
438			0	M	0	1	5842	0	maxExportStack	Database	
439			0	M	0	1	5845	0	ExportStackProps	Database	
440			0	M	0	1	5866	5864	MaxString	Database	
441			0	M	0	1	6112	6110	systemZone	Database	
442		StructChart					6158	706			
443			40	-	0	13	6163	0	created	Formal Para	
444			40	G	1	0	6232	6146	posIndex	Database	
445			40	G	1	0	6236	5870	fullCount	Database	
446			40	G	1	0	6240	6120	positionStackPtr	Database	
447			40	G	0	1	6246	5775	PositionStack	Database	
447			40	G	1	0	6249	6126	fullStackPtr	Database	
448			40	M	0	1	6257	6255	FullStack	Database	
449			40	I	0	1	6261	6173	aIndexPtr	Database	
450			40	E	1	1	6273	6114	statsFile	Database	
451			40	M	0	1	6287	0	text	Database	
452			40	E	1	1	6304	6114	statsFile	Call Param	delete
453			40	I	0	1	6314	6449	aIndexPtr	Call Param	FormatData
454			40	I	0	1	6316	6454	aStackPtr	Call Param	FormatData
455			40	I	0	1	6318	6460	fileStackPtr	Call Param	FormatData
456			40	I	0	1	6320	6466	MaxFiles	Call Param	FormatData
457			40	I	0	1	6322	6470	SCGraph	Call Param	FormatData
458			40	I	0	1	6324	6474	SCTable	Call Param	FormatData
459			40	I	0	1	6326	6478	debugg	Call Param	FormatData
460			40	I	0	1	6328	6482	repeats	Call Param	FormatData
461			40	I	0	1	6332	6198	SCGraph	Database	FormatData
462			40	I	0	1	6334	6202	SCTable	Database	
463			40	I	0	1	6334	6202	SCTable	Database	

VARIABLE DUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
464			40	I	0	1	6357	6173	aIndexPtr	Call Param	DataFlow
465			40	I	0	1	6359	6167	aStackPtr	Call Param	DataFlow
466			40	I	0	1	6361	6179	extSp	Call Param	DataFlow
467			40	I	0	1	6363	6184	maxExt	Call Param	DataFlow
468			40	I	0	1	6365	6188	fileStackPtr	Call Param	DataFlow
469			40	I	0	1	6367	6194	MaxFiles	Call Param	DataFlow
470			40	G	0	2	6369	6120	positionStackPtr	Call Param	DataFlow
471			40	G	0	2	6371	6126	fullStackPtr	Call Param	DataFlow
472			40	G	0	1	6373	5870	fullCount	Call Param	DataFlow
473			40	I	0	1	6375	6206	DFDVariable	Call Param	DataFlow
474			40	I	0	1	6377	6210	DFDTable	Call Param	DataFlow
475			40	I	0	1	6379	6214	debugg	Call Param	DataFlow
476			40	E	0	1	6388	6114	statsFile	Call Param	DataFlow
477			40	E	0	1	6394	6114	statsFile	Call Param	SetLength
478			40	E	1	1	6402	6114	statsFile	Call Param	GetPosition
479			40	L	1	0	6409	6228	result	DataBase	delete
480			40	L	0	1	6440	6228	result	Return Valu	
481		FormatData					6445	0			
482			41	G	4	5	6542	6140	stackIndex	DataBase	
483			41	G	1	0	6546	6152	lineCount	DataBase	
484			41	G	4	4	6550	6146	posIndex	DataBase	
485			41	I	0	2	6555	6470	SCGraph	DataBase	
486			41	I	0	2	6557	6474	SCTable	DataBase	
487			41	I	0	1	6562	9444	fileStackPtr	Call Param	WriteFiles
488			41	I	0	1	6564	9450	MaxFiles	Call Param	WriteFiles
489			41	I	0	1	6576	9895	aIP	Call Param	WriteExportInfo
490			41	I	0	1	6578	9900	atomStackPtr	Call Param	WriteExportInfo
491			41	I	0	1	6580	9906	fileStackPtr	Call Param	WriteExportInfo
492			41	I	0	1	6582	9912	MaxFiles	Call Param	WriteExportInfo
493			41	I	0	1	6608	6460	fileStackPtr	DataBase	
494			41	D	0	1	6613	6608	endIndex	DataBase	
495			41	I	0	1	6631	8758	atomStackPtr	Call Param	CheckReturnCondition
496			41	I	0	1	6633	8764	debugg	Call Param	CheckReturnCondition
497			41	I	0	1	6637	6454	atomStackPtr	DataBase	
498			41	D	0	1	6642	6637	idMark	DataBase	
499			41	I	0	1	6656	6955	atomStackPtr	Call Param	CheckIfTerminalDecl
500			41	I	0	1	6658	6961	fileStackPtr	Call Param	CheckIfTerminalDecl
501			41	L	4	0	6662	6536	SLC	DataBase	
502			41	I	0	1	6678	7062	atomStackPtr	Call Param	CollectPositionData
503			41	I	0	1	6680	7068	debugg	Call Param	CollectPositionData
504			41	I	0	1	6685	7868	atomStackPtr	Call Param	UpdateFullStack
505			41	I	0	1	6687	7874	fileStackPtr	Call Param	UpdateFullStack
506			41	L	0	1	6689	7880	SLC	Call Param	UpdateFullStack
507			41	L	0	1	6691	7884	found	Call Param	UpdateFullStack
508			41	I	0	1	6703	8324	atomStackPtr	Call Param	SuppressOutput
509			41	I	0	1	6705	8330	debugg	Call Param	SuppressOutput
510			41	I	0	1	6715	8324	atomStackPtr	Call Param	SuppressOutput
511			41	I	0	1	6717	8330	debugg	Call Param	SuppressOutput
512			41	I	0	1	6748	7062	atomStackPtr	Call Param	CollectPositionData
513			41	I	0	1	6750	7068	debugg	Call Param	CollectPositionData
514			41	I	0	1	6755	7868	atomStackPtr	Call Param	UpdateFullStack
515			41	I	0	1	6757	7874	fileStackPtr	Call Param	UpdateFullStack

VARIABLE OUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
516			41	L	0	1	6759	7880	SLC	Call Param	UpdateFullStack
517			41	L	0	1	6761	7884	found	Call Param	UpdateFullStack
518			41	L	1	2	6769	6530	found	DataBase	
519			41	I	0	1	6774	8491	atomStackPtr	Call Param	SearchForDeclaration
520			41	I	0	1	6776	8497	fileStackPtr	Call Param	SearchForDeclaration
521			41	I	0	1	6778	8503	MaxFiles	Call Param	SearchForDeclaration
522			41	I	0	1	6787	7062	atomStackPtr	Call Param	CollectPositionData
523			41	I	0	1	6789	7068	debugg	Call Param	CollectPositionData
524			41	I	0	1	6794	7868	atomStackPtr	Call Param	UpdateFullStack
525			41	I	0	1	6796	7874	fileStackPtr	Call Param	UpdateFullStack
526			41	L	0	1	6798	7880	SLC	Call Param	UpdateFullStack
527			41	L	0	1	6800	7884	found	Call Param	UpdateFullStack
528			41	I	0	1	6805	8043	aIP	Call Param	GoFindCallParams
529			41	I	0	1	6807	8049	atomStackPtr	Call Param	GoFindCallParams
530			41	G	0	1	6809	8055	fullStackPtr	Call Param	GoFindCallParams
531			41	I	0	1	6831	6449	aIP	Call Param	GoFindCallParams
532			41	I	0	1	6854	9161	atomStackPtr	DataBase	StudyRepeats
533			41	I	0	1	6874	7319	atomStackPtr	Call Param	SemiGraphOutput
534			41	I	0	1	6876	7325	fileStackPtr	Call Param	SemiGraphOutput
535			41	I	0	1	6878	7331	debugg	Call Param	SemiGraphOutput
536			41	I	0	1	6880	7335	repeats	Call Param	SemiGraphOutput
537			41	I	0	1	6897	11128	atomStackPtr	Call Param	TabularOutput
538			41	I	0	1	6899	11134	fileStackPtr	Call Param	TabularOutput
539			41	I	0	1	6901	11140	debugg	Call Param	TabularOutput
540			41	I	0	1	6903	11144	repeats	Call Param	TabularOutput
541		CleanUpPositionData					6908	0			
542			42	G	0	4	6913	6120	positionStackPtr	DataBase	
543			42	G	0	4	6915	6140	stackIndex	DataBase	
544			42	D	1	0	6918	6913	declLocation	DataBase	
545			42	D	1	0	6927	6922	dProcLevel	DataBase	
546			42	D	1	0	6936	6931	callLocation	DataBase	
547			42	D	1	0	6945	6940	procLevel	DataBase	
548		CheckIfTerminalDecl					6951	0			
549			43	L	1	4	6992	6973	tempIndex	DataBase	
550			43	I	0	1	6994	6961	fSP	DataBase	
551			43	L	0	1	6996	6979	tempFileCount	DataBase	
552			43	D	0	1	6999	6994	endIndex	DataBase	
553			43	I	0	3	7008	6955	aSP	DataBase	
554			43	D	0	1	7013	7008	idMark	DataBase	
555			43	L	0	1	7015	6524	procCall	DataBase	
556			43	M	0	1	7021	7019	Equal	DataBase	
557			43	G	0	1	7025	6146	posIndex	DataBase	
558			43	D	0	1	7028	7023	inputString	DataBase	
559			43	D	0	1	7035	7030	inputString	DataBase	
560			43	L	1	0	7039	6985	terminal	DataBase	
561		CollectPositionData					7053	6985	terminal	Return Valu	
562							7058	0			
563			44	I	0	3	7075	7062	aSP	DataBase	
564			44	G	0	5	7077	6146	posIndex	DataBase	
565			44	D	0	1	7080	7075	idMark	DataBase	
566			44	L	0	1	7082	6518	procedure	DataBase	
567			44	G	0	8	7085	6120	positionStackPtr	DataBase	

VARIABLE OUPUT TABLE

vSp Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
568			44	G	0	8	7087	6140	stackIndex	Database	
569			44	D	1	0	7090	7085	declLocation	Database	
570			44	D	1	0	7099	7094	dprocLevel	Database	
571			44	D	0	1	7106	7101	Plevel	Database	
572			44	D	1	0	7116	7111	callLocation	Database	
573			44	D	1	0	7125	7120	procLevel	Database	
574			44	D	0	1	7132	7127	Plevel	Database	
575			44	I	0	1	7137	7068	debugg	Database	
576			44	G	1	0	7145	6098	numString	Database	
577			44	M	0	4	7149	7147	CopyToNewString	Database	
578			44	M	0	4	7153	7151	CardToCString	Database	
579			44	D	0	1	7160	7155	callLocation	Database	
580			44	G	0	1	7163	6105	z	Database	
581			44	G	0	1	7168	10789	numString	Call Param	StatsWrite
582			44	G	0	2	7175	6105	z	Call Param	FreeString
583			44	G	1	1	7177	6098	numString	Call Param	FreeString
584			44	D	0	1	7200	7195	procLevel	Database	
585			44	G	0	1	7208	10789	numString	Call Param	StatsWrite
586			44	G	0	2	7215	6105	z	Call Param	FreeString
587			44	G	1	1	7217	6098	numString	Call Param	FreeString
588			44	D	0	1	7240	7235	declLocation	Database	
589			44	G	0	1	7248	10789	numString	Call Param	StatsWrite
590			44	G	0	2	7255	6105	z	Call Param	FreeString
591			44	G	1	1	7257	6098	numString	Call Param	FreeString
592			44	D	0	1	7280	7275	dprocLevel	Database	
593			44	G	0	1	7288	10789	numString	Call Param	StatsWrite
594			44	G	0	1	7295	6105	z	Call Param	FreeString
595			44	G	0	1	7297	6098	numString	Call Param	FreeString
596			44	E	0	1	7304	6114	statsFile	Call Param	PutChar
597			44	M	0	1	7308	7306	CR	Call Param	PutChar
598		SemiGraphOutput					7315	0			
599			45	L	1	6	7400	7357	tempCount	Database	
600			45	G	0	2	7402	5870	fullCount	Database	
601			45	L	2	2	7410	7351	tempLevel	Database	
602			45	L	1	4	7414	7369	currentCall	Database	
603			45	G	0	6	7416	6126	fullStackPtr	Database	
604			45	D	0	1	7421	7416	callLocation	Database	
605			45	L	1	0	7423	7387	currentLevel	Database	
606			45	D	0	1	7430	7425	callLevel	Database	
607			45	L	1	1	7432	7393	SLC	Database	
608			45	D	0	1	7439	7434	SLC	Database	
609			45	D	0	1	7448	7443	twinRepeat	Database	
610			45	I	0	1	7450	7335	repeats	Database	
611			45	G	1	0	7453	6098	numString	Database	
612			45	L	1	5	7463	7357	tempCount	Call Param	CardToCString
613			45	G	0	1	7466	6105	z	Database	
614			45	G	0	1	7471	10811	maxIndexLength	Call Param	CheckLength
615			45	G	0	1	7476	10789	numString	Call Param	StatsWrite
616			45	G	0	2	7483	6105	z	Call Param	FreeString
617			45	G	1	1	7485	6098	numString	Call Param	FreeString
618			45	L	2	2	7493	7363	fileNumber	Database	
619			45	D	0	1	7500	7495	fileNumber	Database	

VARIABLE OUTPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
620			45	M	0	4	7506	7504	CopyToNewString	DataBase	
621			45	I	0	2	7508	7325	fSP	DataBase	
622			45	D	0	1	7513	7508	fileName	DataBase	
623			45	G	0	1	7520	10811	maxFileLength	Call Param	CheckLength
624			45	G	0	1	7525	10789	numString	Call Param	StatWrite
625			45	G	0	2	7532	6105	z	Call Param	FreeString
626			45	G	1	1	7534	6098	numString	Call Param	FreeString
627			45	I	0	4	7548	7319	aSP	DataBase	
628			45	D	0	1	7553	7548	IEatom	DataBase	
629			45	G	0	1	7560	10811	maxIIdentLength	Call Param	CheckLength
630			45	G	0	1	7565	10789	numString	Call Param	StatWrite
631			45	G	0	1	7572	6105	z	Call Param	FreeString
632			45	G	2	1	7574	6098	numString	Call Param	FreeString
633			45	L	1	1	7582	7375	currentDecl	DataBase	
634			45	D	0	1	7589	7584	declLocation	DataBase	
635			45	L	1	1	7591	7341	deref	DataBase	
636			45	D	0	1	7598	7593	varScope	DataBase	
637			45	G	0	2	7618	6105	z	Call Param	CopyToNewString
638			45	I	0	1	7628	9083	fSP	Call Param	FindFileNumber
639			45	L	0	1	7630	9089	currentDecl	Call Param	FindFileNumber
640			45	D	0	1	7644	7639	fileName	DataBase	
641			45	G	0	1	7653	10811	maxFileLength	Call Param	CheckLength
642			45	G	0	1	7658	10789	numString	Call Param	StatWrite
643			45	G	0	2	7665	6105	z	Call Param	FreeString
644			45	G	1	1	7667	6098	numString	Call Param	FreeString
645			45	L	0	3	7685	7387	currentLevel	Call Param	CardToString
646			45	G	0	1	7693	10789	numString	Call Param	StatWrite
647			45	G	0	2	7700	6105	z	Call Param	FreeString
648			45	G	1	1	7702	6098	numString	Call Param	FreeString
649			45	D	0	1	7752	7747	inputString	DataBase	
650			45	G	0	1	7759	10789	numString	Call Param	StatWrite
651			45	G	0	1	7766	6105	z	Call Param	FreeString
652			45	G	0	1	7768	6098	numString	Call Param	FreeString
653			45	E	0	1	7775	6114	statsFile	Call Param	PutChar
654			45	M	0	1	7779	7777	CR	Call Param	PutChar
655			45	G	2	4	7782	6152	lineCount	DataBase	
656			45	G	0	2	7791	5888	pageSize	DataBase	
657			45	D	0	1	7815	7808	callLevel	DataBase	
658			45	E	0	1	7853	6114	statsFile	Call Param	PutChar
659			45	M	0	1	7857	7855	CR	Call Param	PutChar
660		UpdateFullStack					7864	0			
661			46	G	2	9	7904	5870	fullCount	DataBase	
662			46	M	0	1	7908	7906	maxStack	DataBase	
663			46	L	1	0	7911	7890	count	DataBase	
664			46	M	0	1	7919	7917	maxStack	Call Param	CardToString
665			46	L	0	1	7933	7890	count	Call Param	log
666			46	I	0	1	7952	7884	found	DataBase	
667			46	G	0	6	7955	6126	fullStackPtr	DataBase	
668			46	D	1	0	7960	7955	declLocation	DataBase	
669			46	G	0	1	7962	6120	positionStackPtr	DataBase	
670			46	G	0	2	7964	6140	stackIndex	DataBase	
671			46	D	0	1	7967	7962	declLocation	DataBase	

VARIABLE DUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
672			46	L	1	1	7978	7897	fileNumber	Database	
673			46	I	0	1	7982	9083	fSP	Call Param	FindFileNumber
674			46	G	0	1	7984	9089	posIndex	Call Param	FindFileNumber
675			46	D	1	0	7992	7987	callLocation	Database	
676			46	G	0	2	7994	6146	posIndex	Database	
677			46	D	1	0	8001	7996	callLevel	Database	
678			46	D	1	0	8010	8005	fileNumber	Database	
679			46	I	0	1	8015	7880	SLC	Database	
680			46	D	1	0	8022	8017	SLC	Database	
681			46	D	1	0	8031	8026	declLocation	Database	
682		GoFindCallParams					8039	0			
683			47	L	3	10	8093	8063	callIndex	Database	
684			47	I	0	2	8095	8055	fullStackPtr	Database	
685			47	G	0	2	8097	5870	fullCount	Database	
686			47	D	0	1	8100	8095	callLocation	Database	
687			47	I	0	13	8103	8049	asP	Database	
688			47	D	0	1	8108	8103	listMark	Database	
689			47	G	0	2	8110	6056	callParameter	Database	
690			47	D	0	1	8124	8119	symbolNumber	Database	
691			47	G	0	4	8126	6086	semiColon	Database	
692			47	D	0	1	8136	8131	symbolNumber	Database	
693			47	G	0	2	8138	6032	equals	Database	
694			47	D	0	1	8150	8145	listMark	Database	
695			47	D	0	1	8166	8161	symbolNumber	Database	
696			47	D	0	1	8180	8173	symbolNumber	Database	
697			47	D	0	1	8192	8187	idMark	Database	
698			47	G	0	2	8194	6092	variable	Database	
699			47	L	1	2	8197	8081	paramCount	Database	
700			47	L	3	8	8203	8069	declIndex	Database	
701			47	D	0	1	8210	8205	declLocation	Database	
702			47	L	2	2	8212	8075	declParamCount	Database	
703			47	D	0	1	8222	8217	listMark	Database	
704			47	G	0	2	8224	6062	parameter	Database	
705			47	D	0	1	8238	8233	symbolNumber	Database	
706			47	D	0	0	8252	8247	listMark	Database	
707			47	D	0	1	8268	8263	symbolNumber	Database	
708			47	D	0	1	8280	8275	idMark	Database	
709			47	D	1	0	8302	8297	declLocation	Database	
710		SuppressOutput					8320	0			
711			48	G	0	1	8340	6146	posIndex	Database	
712			48	L	1	1	8346	8342	startingPLevel	Database	
713			48	I	0	2	8348	8324	asP	Database	
714			48	L	1	4	8350	8336	tempIndex	Database	
715			48	D	0	1	8353	8348	plevel	Database	
716			48	I	0	2	8356	8330	debugg	Database	
717			48	G	1	0	8364	6098	numString	Database	
718			48	G	1	1	8374	6146	posIndex	Call Param	CardToString
719			48	G	0	1	8377	6105	z	Database	
720			48	G	0	1	8382	10811	maxIndexLength	Call Param	CheckLength
721			48	G	0	1	8387	10789	numString	Call Param	StatsWrite
722			48	G	0	2	8394	6105	z	Call Param	FreeString
723			48	G	1	1	8396	6098	numString	Call Param	FreeString

VARIABLE OUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
724			48	D	0	1	8412	8407	Plevel	DataBase	
725			48	G	0	1	8447	6146	posIndex	Call Param	CardIoString
726			48	G	0	1	8455	10811	maxIndexLength	Call Param	CheckLength
727			48	G	0	1	8460	10789	numString	Call Param	StatsWrite
728			48	G	0	1	8467	6105	z	Call Param	FreeString
729			48	G	0	1	8469	6098	numString	Call Param	FreeString
730			48	E	0	1	8476	6114	statsFile	Call Param	PutChar
731			48	M	0	1	8480	8478	CR	Call Param	PutChar
732		SearchForDeclaration					8487	0			
733			49	L	3	6	8543	8525	tempFileCount	DataBase	
734			49	I	0	1	8550	8503	MaxFiles	DataBase	
735			49	I	0	4	8559	8497	fSP	DataBase	
736			49	D	0	1	8564	8559	fileCount	DataBase	
737			49	M	0	2	8573	8571	Equal	DataBase	
738			49	I	0	7	8575	8491	aSP	DataBase	
739			49	G	1	5	8577	6146	posIndex	DataBase	
740			49	D	0	1	8580	8575	IEAtom	DataBase	
741			49	D	0	1	8587	8582	exportName	DataBase	
742			49	G	1	0	8591	6098	numString	DataBase	
743			49	M	0	1	8595	8593	CopyToNewString	DataBase	
744			49	D	0	1	8602	8597	inputString	DataBase	
745			49	G	0	1	8604	6105	z	DataBase	
746			49	L	1	3	8608	8519	found	DataBase	
747			49	I	0	1	8613	10996	aSP	Call Param	CheckProcExists
748			49	I	0	1	8615	11002	fSP	Call Param	CheckProcExists
749			49	L	0	1	8617	11008	tempFileCount	Call Param	CheckProcExists
750			49	G	0	1	8619	11012	numString	Call Param	CheckProcExists
751			49	G	0	1	8626	6105	z	Call Param	FreeString
752			49	G	0	1	8628	6098	numString	Call Param	FreeString
753			49	L	1	0	8640	8513	callLocation	DataBase	
754			49	D	0	1	8647	8642	Index	DataBase	
755			49	I	0	1	8653	9083	fSP	Call Param	FindFileNumber
756			49	L	0	1	8655	9089	callLocation	Call Param	FindFileNumber
757			49	L	2	6	8658	8531	tempIndex	DataBase	
758			49	D	0	1	8665	8660	startIndex	DataBase	
759			49	D	0	1	8680	8675	endIndex	DataBase	
760			49	D	0	1	8694	8689	idMark	DataBase	
761			49	L	0	1	8696	6518	procedure	DataBase	
762			49	D	0	1	8709	8704	inputString	DataBase	
763			49	D	0	1	8716	8711	inputString	DataBase	
764			49	D	1	0	8725	8720	declLocation	DataBase	
765		CheckReturnCondition	49	L	0	1	8749	8519	found	Return Valu	
766							8754	0			
767			50	G	0	7	8771	6120	positionStackPtr	DataBase	
768			50	G	3	11	8773	6140	stackIndex	DataBase	
769			50	D	0	1	8776	8771	procLevel	DataBase	
770			50	D	0	1	8785	8780	callLocation	DataBase	
771			50	D	0	1	8796	8791	dProcLevel	DataBase	
772			50	I	0	3	8798	8758	aSP	DataBase	
773			50	G	2	2	8800	6146	posIndex	DataBase	
774			50	D	0	1	8803	8798	Plevel	DataBase	
775			50	D	0	1	8813	8808	callLocation	DataBase	

VARIABLE OUTPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
776			50	I	0	3	8828	8764	debugg	Database	
777			50	G	1	0	8836	6098	numString	Database	
778			50	G	0	2	8846	6146	posIndex	Call Param	CardToString
779			50	G	0	1	8849	6105	z	Database	
780			50	G	0	1	8854	10811	maxIndexLength	Call Param	CheckLength
781			50	G	0	1	8859	10789	numString	Call Param	StatsWrite
782			50	G	0	2	8866	6105	z	Call Param	FreeString
783			50	G	1	1	8868	6098	numString	Call Param	FreeString
784			50	E	0	1	8875	6114	statsFile	Call Param	PutChar
785			50	M	0	1	8879	8877	CR	Call Param	PutChar
786			50	D	0	1	8901	8896	dProcLevel	Database	
787			50	D	0	1	8908	8903	pLevel	Database	
788			50	G	2	3	8936	6146	posIndex	Call Param	CardToString
789			50	G	0	1	8944	10811	maxIndexLength	Call Param	CheckLength
790			50	G	0	1	8949	10789	numString	Call Param	StatsWrite
791			50	G	0	2	8956	6105	z	Call Param	FreeString
792			50	G	1	1	8958	6098	numString	Call Param	FreeString
793			50	E	0	1	8965	6114	statsFile	Call Param	PutChar
794			50	M	0	1	8969	8967	CR	Call Param	PutChar
795			50	D	0	1	8984	8979	dProcLevel	Database	
796			50	D	0	1	8991	8986	pLevel	Database	
797			50	D	0	1	9001	8996	callLocation	Database	
798			50	G	0	1	9034	6146	posIndex	Call Param	CardToString
799			50	G	0	1	9042	10811	maxIndexLength	Call Param	CheckLength
800			50	G	0	1	9047	10789	numString	Call Param	StatsWrite
801			50	G	0	1	9054	6105	z	Call Param	FreeString
802			50	G	0	1	9056	6098	numString	Call Param	FreeString
803			50	E	0	1	9063	6114	statsFile	Call Param	PutChar
804			50	M	0	1	9067	9065	CR	Call Param	PutChar
805		FindFileNumber	51	L	1	1	9112	9105	notFound	Database	
806			51	I	0	2	9115	9089	callLocation	Database	
807			51	I	0	2	9117	9083	fSP	Database	
808			51	L	1	3	9119	9099	fileCount	Database	
809			51	D	0	1	9122	9117	startIndex	Database	
810			51	D	0	1	9131	9126	andIndex	Database	
811			51	L	0	1	9152	9099	fileCount	Return Valu	
812			51	L	0	1	9157	0			
813		StudyRepeats	52	L	1	9	9206	9169	mainLoopCount	Database	
814			52	G	0	1	9208	5870	fullCount	Database	
815			52	L	1	2	9216	9181	currentCall	Database	
816			52	G	0	13	9218	6126	fullStackPtr	Database	
817			52	D	0	1	9223	9218	callLocation	Database	
818			52	L	1	3	9225	9193	currentLevel	Database	
819			52	D	0	1	9232	9227	callLevel	Database	
820			52	L	4	12	9234	9175	localLoopCount	Database	
821			52	D	0	1	9255	9250	callLevel	Database	
822			52	D	0	1	9267	9262	callLevel	Database	
823			52	D	0	2	9272	9187	comparisonCall	Database	
824			52	L	2	2	9279	9274	callLocation	Database	
825			52	M	0	2	9284	9282	Equal	Database	
826			52	I	0	4	9286	9161	aSP	Database	
827			52	I	0	4	9286	9161	aSP	Database	

VARIA8LE OUPUT TA8LE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
828			52	D	0	1	9291	9288	inputString	Database	
829			52	D	0	0	9298	9293	inputString	Database	
830			52	D	1	0	9307	9302	twinRepeat	Database	
831			52	D	1	0	9316	9311	firstOfTwin	Database	
832			52	D	0	0	9354	9349	callLocation	Database	
833			52	L	1	1	9356	9199	comparisonLevel	Database	
834			52	D	0	1	9363	9358	callLevel	Database	
835			52	D	0	1	9375	9370	inputString	Database	
836			52	D	0	1	9382	9377	inputString	Database	
837			52	D	1	0	9391	9386	globalRepeat	Database	
838			52	D	1	0	9400	9395	firstOfGlobal	Database	
839			52	D	1	0	9415	9410	levelRepeat	Database	
840			52	D	1	0	9424	9419	firstOfLevel	Database	
841		WriteFiles					9440	0			
842			53	E	0	1	9507	6114	statsFile	Call Param	PutChar
843			53	M	0	1	9511	9509	CR	Call Param	PutChar
844			53	E	0	1	9523	6114	statsFile	Call Param	PutChar
845			53	M	0	1	9527	9525	CR	Call Param	PutChar
846			53	E	0	1	9534	6114	statsFile	Call Param	PutChar
847			53	M	0	1	9538	9536	CR	Call Param	PutChar
848			53	E	0	1	9550	6114	statsFile	Call Param	PutChar
849			53	M	0	1	9554	9552	CR	Call Param	PutChar
850			53	E	0	1	9561	6114	statsFile	Call Param	PutChar
851			53	M	0	1	9565	9563	CR	Call Param	PutChar
852			53	E	0	1	9577	6114	statsFile	Call Param	PutChar
853			53	M	0	1	9581	9579	CR	Call Param	PutChar
854			53	E	0	1	9593	6114	statsFile	Call Param	PutChar
855			53	M	0	1	9597	9595	CR	Call Param	PutChar
856			53	G	3	4	9600	6152	lineCount	Database	
857			53	L	0	1	9604	9492	titleSize	Database	
858			53	L	1	8	9607	9486	tempIndex	Database	
859			53	I	0	1	9609	9450	MaxFiles	Database	
860			53	I	0	6	9624	9444	fileStackPtr	Database	
861			53	D	0	1	9629	9624	fileCount	Database	
862			53	G	1	0	9640	6098	numString	Database	
863			53	M	0	5	9644	9642	CopyToNewString	Database	
864			53	M	0	3	9648	9646	CardToString	Database	
865			53	D	0	1	9655	9650	fileCount	Database	
866			53	G	0	1	9658	6105	z	Database	
867			53	L	0	1	9663	10811	maxFileIndexLength	Call Param	CheckLength
868			53	G	0	1	9668	10789	numString	Call Param	StatsWrite
869			53	G	0	2	9675	6105	z	Call Param	FreeString
870			53	G	1	1	9677	6098	numString	Call Param	FreeString
871			53	D	0	1	9696	9691	fileName	Database	
872			53	L	0	1	9703	10811	maxFileNameLength	Call Param	CheckLength
873			53	G	0	1	9708	10789	numString	Call Param	StatsWrite
874			53	G	0	2	9715	6105	z	Call Param	FreeString
875			53	G	1	1	9717	6098	numString	Call Param	FreeString
876			53	D	0	1	9740	9735	startIndex	Database	
877			53	L	0	1	9748	10811	maxStartLength	Call Param	CheckLength
878			53	G	0	1	9753	10789	numString	Call Param	StatsWrite
879			53	G	0	2	9760	6105	z	Call Param	FreeString

VARIABLE OUPUT TABLE

VSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
880			53	G	1	1	9762	6098	numString	Call Param	FreeString
881			53	D	0	1	9785	9780	endIndex	Database	
882			53	L	0	1	9793	10811	maxEndLength	Call Param	CheckLength
883			53	G	0	1	9798	10789	numString	Call Param	StatsWrite
884			53	G	0	2	9805	6105	z	Call Param	FreeString
885			53	G	1	1	9807	6098	numString	Call Param	FreeString
886			53	D	0	1	9826	9821	exportName	Database	
887			53	L	0	1	9833	10811	maxExportNameLength	Call Param	CheckLength
888			53	G	0	1	9838	10789	numString	Call Param	StatsWrite
889			53	G	0	1	9845	6105	z	Call Param	FreeString
890			53	G	0	1	9847	6098	numString	Call Param	FreeString
891			53	E	0	1	9864	6114	statsFile	Call Param	FreeString
892			53	M	0	1	9858	9856	CR	Call Param	PutChar
893			53	G	0	1	9866	5888	pageSize	Database	PutChar
894			53	E	0	1	9880	6114	statsFile	Call Param	PutChar
895			53	M	0	1	9884	9882	CR	Call Param	PutChar
896		WriteExportInfo					9891	0			
897			54	G	1	4	9972	6134	exportStackPtr	Database	
898			54	G	0	1	9978	5834	ExportStack	Database	
899			54	L	1	7	9982	9930	tempIndex	Database	
900			54	I	0	1	9984	9895	aIP	Database	
901			54	I	0	7	9994	9900	aSP	Database	
902			54	D	0	1	9999	9994	idMark	Database	
903			54	L	0	1	10001	9948	procCall	Database	
904			54	M	0	3	10008	10006	Equal	Database	
905			54	D	0	1	10015	10010	IEatom	Database	
906			54	L	2	4	10021	9936	tempFileCount	Database	
907			54	I	0	1	10028	9912	MaxFiles	Database	
908			54	I	0	2	10037	9906	fSP	Database	
909			54	D	0	1	10042	10037	fileCount	Database	
910			54	D	0	1	10058	10053	IEatom	Database	
911			54	D	0	1	10065	10060	exportName	Database	
912			54	G	1	0	10069	6098	numString	Database	
913			54	M	0	1	10073	10071	CopyToNewString	Database	
914			54	D	0	1	10080	10075	inputString	Database	
915			54	G	0	1	10082	6105	z	Database	
916			54	L	1	3	10086	9942	DK	Database	
917			54	I	0	1	10091	10996	aSP	Call Param	CheckProcExists
918			54	I	0	1	10093	11002	fSP	Call Param	CheckProcExists
919			54	L	0	1	10095	11008	tempFileCount	Call Param	CheckProcExists
920			54	G	0	1	10097	11012	numString	Call Param	CheckProcExists
921			54	G	0	1	10104	6105	z	Call Param	FreeString
922			54	G	0	1	10106	6098	numString	Call Param	FreeString
923			54	L	2	5	10127	9960	exportStackCount	Database	
924			54	D	1	0	10138	10133	missingProcLocation	Database	
925			54	D	0	1	10145	10140	Index	Database	
926			54	L	2	3	10147	9966	loopCount	Database	
927			54	L	1	1	10162	9918	currentCall	Database	
928			54	D	0	1	10169	10164	missingProcLocation	Database	
929			54	L	1	1	10171	9924	lastCall	Database	
930			54	D	0	1	10178	10173	missingProcLocation	Database	
931			54	D	0	1	10190	10185	inputString	Database	

VARIA8LE OUPUT TAB8LE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
932	OutputExportInfo	OutputExportInfo	54	D	0	1	10197	10192	inputString	Database	OutputExportInfo OutputExportInfo OutputExportInfo OutputExportInfo
933			54	L	0	1	10223	10246	exportStackCount	Call Param	
934			54	G	0	1	10225	10250	exportStackPtr	Call Param	
935			54	I	0	1	10227	10257	fSP	Call Param	
936			54	I	0	1	10229	10263	aSP	Call Param	
937							10242	0			
938			55	G	4	6	10305	6152	lineCount	Database	
939			55	L	0	2	10309	10295	titleSize	Database	
940			55	L	1	2	10312	10289	loopCount	Database	
941			55	I	0	1	10314	10246	eSC	Database	
942			55	G	0	2	10331	5888	pageSize	Database	
943			55	G	1	0	10348	6098	numString	Database	
944			55	L	0	2	10358	10289	loopCount	Call Param	CardToString CheckLength StatsWrite FreeString FreeString
945			55	G	0	1	10361	6105	z	Database	
946			55	G	0	1	10366	10811	maxIndexLength	Call Param	
947			55	G	0	1	10371	10789	numString	Call Param	
948	WriteExportTitles	WriteExportTitles	55	G	0	2	10378	6105	z	Call Param	Equal CheckLength StatsWrite FreeString FreeString
949			55	G	1	1	10380	6098	numString	Call Param	
950			55	L	1	3	10388	10283	tempIndex	Database	
951			55	I	0	1	10390	10250	eSP	Database	
952			55	D	0	1	10395	10390	missingProcLocation	Database	Equal CheckLength StatsWrite FreeString FreeString
953			55	M	0	3	10401	10399	CopyToNewString	Database	
954			55	I	0	3	10403	10263	aSP	Database	
955			55	D	0	1	10408	10403	IEatom	Database	
956			55	G	0	1	10418	6098	numString	Call Param	CheckLength StatsWrite FreeString FreeString
957			55	L	0	1	10430	10811	maxExportNameLength	Call Param	
958			55	G	0	1	10435	10789	numString	Call Param	
959			55	G	0	2	10442	6105	z	Call Param	
960			55	G	1	1	10444	6098	numString	Call Param	CheckLength StatsWrite FreeString FreeString
961			55	I	0	1	10458	10257	fSP	Database	
962			55	D	0	2	10465	10460	fileNumber	Database	
963			55	L	0	1	10475	10811	maxFileNameLength	Call Param	
964	WriteExportTitles	WriteExportTitles	55	G	0	1	10480	10789	numString	Call Param	CheckLength StatsWrite FreeString FreeString
965			55	G	0	2	10487	6105	z	Call Param	
966			55	G	1	1	10489	6098	numString	Call Param	
967			55	D	0	1	10508	10503	inputString	Database	
968			55	L	0	1	10515	10811	maxFileNameLength	Call Param	CheckLength StatsWrite FreeString FreeString
969			55	G	0	1	10520	10789	numString	Call Param	
970			55	G	0	1	10527	6105	z	Call Param	
971			55	G	0	1	10529	6098	numString	Call Param	
972			55	E	0	1	10536	6114	statsFile	Call Param	PutChar PutChar PutChar PutChar PutChar PutChar
973			55	M	0	1	10540	10538	CR	Call Param	
974			55	E	0	1	10562	6114	statsFile	Call Param	
975			55	M	0	1	10566	10564	CR	Call Param	
976							10573	0			PutChar PutChar PutChar PutChar PutChar PutChar
977			56	E	0	1	10587	6114	statsFile	Call Param	
978			56	M	0	1	10591	10589	CR	Call Param	
979			56	E	0	1	10603	6114	statsFile	Call Param	
980	WriteExportTitles	WriteExportTitles	56	M	0	1	10607	10605	CR	Call Param	PutChar PutChar PutChar PutChar PutChar PutChar
981			56	E	0	1	10619	6114	statsFile	Call Param	
982			56	M	0	1	10623	10621	CR	Call Param	
983			56	E	0	1	10635	6114	statsFile	Call Param	

VARIA6LE OUPUT TA6LE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
984			56	M	0	1	10639	10637	CR	Call Param	PutChar
985			56	E	0	1	10651	6114	statsFile	Call Param	PutChar
986			56	M	0	1	10655	10653	CR	Call Param	PutChar
987			56	E	0	1	10667	6114	statsFile	Call Param	PutChar
988			56	M	0	1	10671	10669	CR	Call Param	PutChar
989			56	E	0	1	10683	6114	statsFile	Call Param	PutChar
990			56	M	0	1	10687	10665	CR	Call Param	PutChar
991							10692	0			
992		WriteSCTitles	57	E	0	1	10707	6114	statsFile	Call Param	PutChar
993			57	M	0	1	10711	10709	CR	Call Param	PutChar
994			57	E	0	1	10723	6114	statsFile	Call Param	PutChar
995			57	M	0	1	10727	10725	CR	Call Param	PutChar
996			57	E	0	1	10739	6114	statsFile	Call Param	PutChar
997			57	M	0	1	10743	10741	CR	Call Param	PutChar
998			57	E	0	1	10755	6114	statsFile	Call Param	PutChar
999			57	M	0	1	10759	10757	CR	Call Param	PutChar
1000			57	E	0	1	10771	6114	statsFile	Call Param	PutChar
1001			57	M	0	1	10775	10773	CR	Call Param	PutChar
1002			57	G	1	1	10778	6152	lineCount	Call Param	PutChar
1003			57	L	0	1	10782	10697	titleSize	Database	PutChar
1004							10785	0		Database	
1005		StatsWrite	58	E	0	1	10800	6114	statsFile	Call Param	PutString
1006			58	I	0	1	10602	10789	str	Call Param	PutString
1007							10607	0			
1008		CheckLength	59	L	1	2	10821	10817	size	Database	Length
1009			59	G	0	1	10827	6098	numString	Call Param	
1010			59	I	0	2	10833	10811	length	Database	
1011			59	L	0	1	10837	10660	size	Call Param	AddSpaces
1012			59	I	0	1	10839	10664	length	Call Param	AddSpaces
1013			59	L	0	1	10849	10907	size	Call Param	ReduceLength
1014			59	I	0	1	10851	10911	length	Call Param	ReduceLength
1015		AddSpaces					10656	0			
1016			60	I	2	3	10670	10860	size	Database	
1017			60	I	0	1	10672	10864	length	Database	
1018			60	G	0	1	10886	6098	numString	Call Param	AppendStringAndGrow
1019			60	G	0	1	10890	6105	z	Call Param	AppendStringAndGrow
1020		ReduceLength					10903	0			
1021			61	L	1	0	10930	10917	subStringDescr	Database	
1022			61	G	0	1	10933	6098	numString	Database	
1023			61	I	0	1	10937	10911	length	Database	
1024			61	L	1	0	10940	10923	tempString	Database	
1025			61	G	0	1	10946	6105	z	Call Param	MakeString
1026			61	I	0	1	10948	10911	length	Call Param	MakeString
1027			61	L	0	1	10955	10923	tempString	Call Param	AppendSubString
1028			61	L	0	1	10958	10917	subStringDescr	Call Param	AppendSubString
1029			61	G	0	1	10965	6105	z	Call Param	FreeString
1030			61	G	1	1	10967	6098	numString	Call Param	FreeString
1031			61	L	0	1	10976	10923	tempString	Call Param	CopyToNewString
1032			61	G	0	1	10978	6105	z	Call Param	CopyToNewString
1033			61	G	0	1	10985	6105	z	Call Param	FreeString
1034			61	L	0	1	10987	10923	tempString	Call Param	FreeString
1035		CheckProcExists					10992	0			

VARIABLE DUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
1036			62	L	2	6	11041	11023	tempIndex	DataBase	
1037			62	I	0	2	11043	11002	fSP	DataBase	
1038			62	I	0	2	11045	11008	tempfileCount	DataBase	
1039			62	D	0	1	11048	11043	startIndex	DataBase	
1040			62	D	0	1	11058	11053	endIndex	DataBase	
1041			62	I	0	3	11061	10996	aSP	DataBase	
1042			62	D	0	1	11066	11061	idMark	DataBase	
1043			62	L	0	1	11068	11029	proc	DataBase	
1044			62	M	0	1	11074	11072	Equal	DataBase	
1045			62	I	0	1	11076	11012	numString	DataBase	
1046			62	D	0	1	11083	11078	inputString	DataBase	
1047			62	L	1	0	11087	11035	foundProcDeclaration	DataBase	
1048			62	G	1	1	11093	6146	posIndex	DataBase	
1049			62	D	1	0	11096	11091	declLocation	DataBase	
1050			62	L	0	1	11119	11035	foundProcDeclaration	Return Valu	
1051		TabularOutput					11124	0			
1052			63	G	2	3	11208	6152	lineCount	DataBase	
1053			63	L	0	1	11212	11192	titleSize	DataBase	
1054			63	L	3	9	11215	11150	tempStackCount	DataBase	
1055			63	G	0	2	11217	5870	fullCount	DataBase	
1056			63	G	0	5	11220	6126	fullStackPtr	DataBase	
1057			63	D	0	1	11225	11220	SLC	DataBase	
1058			63	L	3	1	11228	11186	SLC	DataBase	
1059			63	L	0	1	11234	11409	tempStackCount	Call Param	OutputTableLine
1060			63	I	0	1	11236	11413	fSP	Call Param	OutputTableLine
1061			63	I	0	1	11238	11419	aSP	Call Param	OutputTableLine
1062			63	L	0	1	11240	11425	SLC	Call Param	OutputTableLine
1063			63	L	0	1	11242	11429	titleSize	Call Param	OutputTableLine
1064			63	L	1	3	11260	11156	currentLevel	DataBase	
1065			63	G	0	1	11262	5819	maxLevels	DataBase	
1066			63	D	0	1	11286	11281	SLC	DataBase	
1067			63	D	0	1	11294	11289	callLevel	DataBase	
1068			63	I	0	1	11300	11144	repeats	DataBase	
1069			63	L	0	1	11305	11409	tempStackCount	Call Param	OutputTableLine
1070			63	I	0	1	11307	11413	fSP	Call Param	OutputTableLine
1071			63	I	0	1	11309	11419	aSP	Call Param	OutputTableLine
1072			63	L	0	1	11311	11425	SLC	Call Param	OutputTableLine
1073			63	L	0	1	11313	11429	titleSize	Call Param	OutputTableLine
1074			63	D	0	1	11331	11326	twInRepeat	Call Param	OutputTableLine
1075			63	D	0	1	11341	11336	levelRepeat	DataBase	
1076			63	L	0	1	11346	11409	tempStackCount	DataBase	
1077			63	I	0	1	11348	11413	fSP	Call Param	OutputTableLine
1078			63	I	0	1	11350	11419	aSP	Call Param	OutputTableLine
1079			63	L	0	1	11352	11425	SLC	Call Param	OutputTableLine
1080			63	L	0	1	11354	11429	titleSize	Call Param	OutputTableLine
1081			63	G	0	1	11380	5888	pageSize	Call Param	OutputTableLine
1082			63	E	0	1	11394	6114	statsFile	DataBase	
1083			63	M	0	1	11398	11396	CR	Call Param	PutChar
1084		OutputTableLine					11405	0		Call Param	PutChar
1085			64	I	0	1	11457	11409	tempStackCount	DataBase	
1086			64	G	0	7	11463	6126	fullStackPtr	DataBase	
1087			64	L	0	1	11465	11453	tSC	DataBase	

VARIABLE DUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
1088			64	D	0	1	11468	11463	callLevel	Database	
1110			64	D	0	1	11605	11600	SLC	Database	
1111			64	G	1	0	11613	6098	numString	Database	
1112			64	L	0	3	11623	11459	level	Call Param	CardToString
1113			64	G	0	1	11631	10811	maxLevelLength	Call Param	CheckLength
1114			64	G	0	1	11636	10789	numString	Call Param	StatsWrite
1115			64	G	0	2	11643	6105	z	Call Param	FreeString
1116			64	G	1	1	11645	6098	numString	Call Param	FreeString
1117			64	L	2	4	11659	11435	callLocation	Database	
1118			64	D	0	1	11666	11661	callLocation	Database	
1119			64	M	0	6	11672	11670	CopyToNewString	Database	
1120			64	I	0	4	11674	11419	asp	Database	
1121			64	D	0	1	11679	11674	inputString	Database	
1122			64	G	0	1	11686	10811	maxProcLength	Call Param	CheckLength
1123			64	G	0	1	11691	10789	numString	Call Param	StatsWrite
1124			64	G	0	2	11698	6105	z	Call Param	FreeString
1125			64	G	1	1	11700	6098	numString	Call Param	FreeString
1126			64	D	0	1	11719	11714	IEatom	Database	
1127			64	G	0	1	11726	10811	maxIdentLength	Call Param	CheckLength
1128			64	G	0	1	11731	10789	numString	Call Param	StatsWrite
1129			64	G	0	2	11738	6105	z	Call Param	FreeString
1130			64	G	1	1	11740	6098	numString	Call Param	FreeString
1131			64	L	2	2	11748	11441	fileNumber	Database	FindFileNumber
1132			64	I	0	1	11752	9083	fSP	Call Param	FindFileNumber
1133			64	L	0	1	11754	9089	callLocation	Call Param	
1134			64	I	0	2	11763	11413	fSP	Database	
1135			64	D	0	1	11768	11763	fileName	Database	
1136			64	L	2	0	11773	11447	size	Database	
1137			64	G	0	1	11779	6098	numString	Call Param	Length
1138			64	L	0	1	11784	10907	size	Call Param	ReduceLength
1139			64	L	0	1	11786	10911	size	Call Param	ReduceLength
1140			64	G	0	1	11793	10811	maxFileLength	Call Param	CheckLength
1141			64	G	0	1	11800	10789	numString	Call Param	StatsWrite
1142			64	G	0	1	11807	6105	z	Call Param	FreeString
1143			64	G	1	1	11809	6098	numString	Call Param	FreeString
1144			64	D	0	1	11824	11819	callLevel	Database	
1145			64	D	0	1	11836	11831	callLevel	Database	
1146			64	D	0	1	11850	11845	twinRepeat	Database	
1147			64	L	1	1	11855	11470	firstLine	Database	
1148			64	G	4	6	11858	6152	lineCount	Database	
1149			64	E	0	1	11868	6114	statsFile	Call Param	PutChar
1150			64	M	0	1	11872	11870	CR	Call Param	PutChar
1151			64	G	0	2	11878	5888	pageSize	Database	
1152			64	I	0	2	11891	11429	titleSize	Database	
1153			64	G	0	1	11903	6105	z	Call Param	CopyToNewString
1154			64	G	0	1	11908	10811	maxSubMargin	Call Param	CheckLength
1155			64	G	0	1	11913	10789	numString	Call Param	StatsWrite
1156			64	G	0	2	11920	6105	z	Call Param	FreeString
1157			64	G	1	1	11922	6098	numString	Call Param	FreeString
1158			64	L	0	1	11937	11459	level	Call Param	CardToString
1159			64	G	0	1	11947	10789	numString	Call Param	StatsWrite
1160			64	G	0	1	11952	10811	maxLevelLength	Call Param	CheckLength

VARIA8LE OUPUT TA8LE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
1161			64	G	0	2	11959	6105	z	Call Param	FreeString
1162			64	G	1	1	11961	6098	numString	Call Param	FreeString
1163			64	D	0	1	11980	11975	callLocation	Database	
1164			64	D	0	1	11993	11988	inputString	Database	
1165			64	G	0	1	12000	10811	maxProcLength	Call Param	CheckLength
1166			64	G	0	1	12005	10789	numString	Call Param	StatsWrite
1167			64	G	0	2	12012	6105	z	Call Param	FreeString
1168			64	G	1	1	12014	6098	numString	Call Param	FreeString
1169			64	D	0	1	12033	12028	IEatom	Database	
1170			64	G	0	1	12040	10811	maxIEidentLength	Call Param	CheckLength
1171			64	G	0	1	12045	10789	numString	Call Param	StatsWrite
1172			64	G	0	2	12052	6105	z	Call Param	FreeString
1173			64	G	1	1	12054	6098	numString	Call Param	FreeString
1174			64	I	0	1	12066	9083	fSP	Call Param	FindFileNumber
1175			64	L	0	1	12068	9089	callLocation	Call Param	FindFileNumber
1176			64	D	0	1	12082	12077	fileName	Call Param	
1177			64	G	0	1	12093	6098	numString	Database	
1178			64	L	0	1	12098	10907	size	Call Param	Length
1179			64	L	0	1	12100	10911	size	Call Param	ReduceLength
1180			64	G	0	1	12107	10811	maxFileLength	Call Param	ReduceLength
1181			64	G	0	1	12114	10789	numString	Call Param	CheckLength
1182			64	G	0	1	12121	6105	z	Call Param	StatsWrite
1183			64	G	0	1	12123	6098	numString	Call Param	FreeString
1184			64	E	0	1	12144	6114	statsFile	Call Param	FreeString
1185			64	M	0	1	12148	12148	CR	Call Param	PutChar
1089		WriteRepeatInfo					11476	0		Call Param	PutChar
1090			65	I	1	0	11481	11012	numString	Database	
1091			65	G	0	1	11489	6105	z	Call Param	CopyToNewString
1092			65	I	0	8	11493	8055	fullStackPtr	Database	
1093			65	L	2	14	11495	11453	tSC	Database	
1094			65	D	0	1	11498	11493	firstOfTwin	Database	
1095			65	D	0	1	11505	11500	twinRepeat	Database	
1096			65	I	0	1	11512	11012	numString	Call Param	AppendStringAndGrow
1097			65	G	0	1	11516	6105	z	Call Param	AppendStringAndGrow
1098			65	D	0	1	11525	11520	firstOfLevel	Database	
1099			65	D	0	1	11532	11527	levelRepeat	Database	
1100			65	I	0	1	11539	11012	numString	Call Param	AppendStringAndGrow
1101			65	G	0	1	11543	6105	z	Call Param	AppendStringAndGrow
1102			65	D	0	1	11552	11547	firstOfGlobal	Database	
1103			65	D	0	1	11559	11554	globalRepeat	Database	
1104			65	I	0	1	11566	11012	numString	Call Param	AppendStringAndGrow
1105			65	G	0	1	11570	6105	z	Call Param	AppendStringAndGrow
1106			65	G	0	1	11575	10811	maxRepeatLength	Call Param	AppendStringAndGrow
1107			65	I	0	1	11580	10789	numString	Call Param	CheckLength
1108			65	G	0	2	11587	6105	z	Call Param	StatsWrite
1109			65	I	0	1	11589	11012	numString	Call Param	FreeString
1186		WriteTableTitles					12179	0		Call Param	FreeString
1187			66	E	0	1	12188	6114	statsFile	Call Param	PutChar
1188			66	M	0	1	12192	12190	CR	Call Param	PutChar
1189			66	E	0	1	12199	6114	statsFile	Call Param	PutChar
1190			66	M	0	1	12203	12201	CR	Call Param	PutChar
1191			66	E	0	1	12215	6114	statsFile	Call Param	PutChar

VARIABLE OUPUT TABLE

vSP Cnt	Program Name	Procedure Name	Proc Count	Var Scope	Write Count	Read Count	Var Loca	Decl Loc	Variable name	Var Type	Procedure Call Name
1192			66	M	0	1	12219	12217	CR	Call Param	PutChar
1193			66	E	0	1	12231	6114	statsFile	Call Param	PutChar
1194			66	M	0	1	12235	12233	CR	Call Param	PutChar
1195			66	E	0	1	12247	6114	statsFile	Call Param	PutChar
1196			66	M	0	1	12251	12249	CR	Call Param	PutChar
1197			66	E	0	1	12263	6114	statsFile	Call Param	PutChar
1198			66	M	0	1	12267	12265	CR	Call Param	PutChar
1199			66	E	0	1	12279	6114	statsFile	Call Param	PutChar
1200			66	M	0	1	12283	12281	CR	Call Param	PutChar
1201			66	E	0	1	12295	6114	statsFile	Call Param	PutChar
1202			66	M	0	1	12299	12297	CR	Call Param	PutChar

APPENDIX A - OUTPUT LISTINGS

A.3

OUTPUT LISTING OF DATAFLOW.DATA

A.3.2 DFD TABLE

```

~ ~ ~ ~ ~ Updating DataBase with DataFlow Info ~ ~ ~ ~ ~
~ ~ ~ ~ ~ Display Complete ~ ~ ~ ~ ~

```

Processing Display ~ ~ ~ ~ ~

- Studying repeat calls.....
- Creating Semi Graphical Output.

Creating DataFlow Chart.....

- Creating variable table.....
- Constructing Context level diagram.
- Constructing level zero diagram. .
- Constructing lower level diagram. .
- Writing dataflow table..

Done Output created to file DataFlow.data

Display Complete

DESIGN EXTRACTION STATISTICS
=====

The files parsed in this session were:

Index	File Name	Start Index	End Index	Exports to
1	DesignExtractorImpl.mesa	1	3778	Utils
2	DisplayImpl.mesa	3779	3945	Display
3	ParserImpl.mesa	3946	5725	Parser
4	StructChartImpl.mesa	5726	12305	Display

DATA FLDW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION						
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias			
2	RunIt	nonT	background	DataBase	Public	0	1	-	-	-			
			priorityBackground	IntCallp	Imported	0	1	Process	ClearIt	Process			
			fileCount	DataBase	Global	1	0	fileCount	DesignExtractorImpl	-			
			fileStackPtr	DataBase	Global	1	1	fileStackPtr	DesignExtractorImpl	-			
			filename	DataBase	Global	0	1	filename	DesignExtractorImpl	-			
			atomStackPtr	DataBase	Global	0	1	FreeAtomStack	AllocateAtomStack	aSP			
			sh	IntCallp	Global	0	1	FreeAtomStack	AllocateAtomStack	sh			
			atomIndexPtr	DataBase	Global	1	1	atomIndexPtr	DesignExtractorImpl	-			
			FileStack	DataBase	Imported	0	1	Parser	-	-			
			AtomIndex	DataBase	Imported	0	1	Parser	-	-			
			sh	DataBase	Global	0	1	sh	DesignExtractorImpl	-			
			atomStackPtr	DataBase	Global	0	1	atomStackPtr	DesignExtractorImpl	-			
			MaxString	IntCallp	Imported	0	1	atomStackPtr	DesignExtractorImpl	-			
			tempZone	DataBase	Public	0	1	AllocateAtomStack	Initialize	elements			
			tempZone	IntCallp	Public	1	1	Heap	ClearIt	tempZone			
			ok	DataBase	Local	1	0	-	-	-			
			ok	IntCallp	Local	0	1	log	Exit	ok			
			2	RunIt	nonT	background	DataBase	Public	0	1	-	-	-
						priorityBackground	IntCallp	Imported	0	1	Process	RunIt	Process
						toolName	IntCallp	Global	0	1	log	Exit	toolName
						totalErrors	DataBase	Global	1	1	totalErrors	DesignExtractorImpl	-
						totalWarnings	DataBase	Global	1	1	totalWarnings	DesignExtractorImpl	-
						tempZone	DataBase	Public	1	1	-	-	-
tempZone	IntCallp	Public				1	1	Heap	RunIt	tempZone			
Error	IntCallp	Imported				0	1	Utils	RunIt	MFile			
Error	IntCallp	Imported				0	1	Utils	RunIt	MStream			
aborted	DataBase	Local				2	2	-	-	-			
initialFileLength	DataBase	Local				1	0	-	-	-			
filename	DataBase	Global				0	1	filename	DesignExtractorImpl	-			
token	DataBase	Local				2	1	-	-	-			
UnterminatedQuote	IntCallp	Imported				0	1	token	RunIt	Token			
s	IntCallp	Local				1	2	String	RunIt	s			
s	IntCallp	Local				0	1	Token	RunIt	s			
s	DataBase	Local				1	0	-	-	-			
s	IntCallp	Local				0	1	Token	RunIt	s			
aborted	IntCallp	Local				0	1	log	Exit	aborted			
log	IntCallp	Imported				0	1	Format	RunIt	-			
fileCount	IntCallp	Global				0	1	Format	RunIt	fileCount			
fileCount	IntCallp	Global				0	1	log	Exit	fileCount			
log	IntCallp	Imported				0	1	log	RunIt	-			
totalErrors	IntCallp	Global	0	1	Format	RunIt	totalErrors						
totalErrors	IntCallp	Global	0	1	Format	RunIt	totalErrors						
log	IntCallp	Imported	0	1	log	RunIt	-						
totalWarnings	IntCallp	Global	0	1	Format	RunIt	totalWarnings						
totalWarnings	IntCallp	Global	0	1	Format	RunIt	totalWarnings						
tempZone	IntCallp	Public	1	1	log	Exit	tempZone						
fileSW	IntCallp	Public	0	1	Heap	RunIt	fileSW						
fileSW	IntCallp	Public	0	1	TextSW	RunIt	fileSW						

2 Init
2 MakeTable
2 Add

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE				DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias	
2	EnumeratedDirectory	Term									
2	AddEnumerating	nonT	name	IntCallP	Local	0	1	fileList	AddEnumerating	name	
2	AllEntryNames	Term	ok	FmlParam	-	0	0	-	-	-	
2	CleanUp	nonT	file	DataBase	External	0	1	file	ProcessFile	-	
			delete	DataBase	Imported	0	1	file	-	-	
			file	IntCallP	External	2	1	file	CleanUp	file	
			ok	IntCallP	UseFmlCal	0	1	log	Exit	ok	
			errors	IntCallP	Local	1	3	log	Exit	errors	
			warnings	IntCallP	Local	0	2	log	Exit	warnings	
			log	DataBase	Imported	0	2	Utils	-	-	
			CardToSting	DataBase	Imported	0	2	Utils	-	-	
			fileStackPtr	DataBase	Global	0	7	fileStackPtr	DesignExtractorImpl	-	
			fileCount	DataBase	Global	0	1	fileCount	DesignExtractorImpl	-	
			endIndex	DataBase	Dereferen	0	1	fileStackPtr	CleanUp	-	
			fileCount	IntCallP	Global	1	10	Utils	CleanUp	fileCount	
			atomStackPtr	DataBase	Global	0	2	atomStackPtr	DesignExtractorImpl	-	
			atomIndexPtr	DataBase	Global	0	2	atomIndexPtr	DesignExtractorImpl	-	
			sCount	DataBase	Dereferen	0	1	atomStackPtr	CleanUp	-	
			totalErrors	DataBase	Global	1	1	totalErrors	DesignExtractorImpl	-	
			totalWarnings	DataBase	Global	1	1	totalWarnings	DesignExtractorImpl	-	
2	Address	Term	window	IntCallP	Public	0	1	ToolWindow	ExecutiveProc	window	
2	CopyToNewString	Term	h	IntCallP	Imported	0	1	Exec	Unload	-	
2	AppendMessage	Term	abort	RetVal	Imported	0	1	Exec	Unload	-	
2	AddCommand	Term	window	DataBase	Public	0	1	-	-	-	
2	ExecutiveProc	nonT	window	IntCallP	Public	1	1	Tool	Unload	window	
			h	IntCallP	Imported	0	1	Exec	Unload	-	
			execCommand	IntCallP	Global	0	1	Exec	Unload	execCommand	
			tempZone	DataBase	Public	0	1	-	-	-	
			tempZone	IntCallP	Public	1	1	Heap	Unload	tempZone	
			permZone	DataBase	Global	0	1	permZone	DesignExtractorImpl	-	
			permZone	IntCallP	Global	1	1	Heap	Unload	permZone	
			atomStackPtr	DataBase	Global	0	1	atomStackPtr	DesignExtractorImpl	-	
			atomStackPtr	IntCallP	Global	0	1	FreeAtomStack	AllocateAtomStack	aSP	
			sh	IntCallP	Global	0	1	FreeAtomStack	AllocateAtomStack	sh	
			fileStackPtr	DataBase	Global	0	2	fileStackPtr	DesignExtractorImpl	-	
			atomIndexPtr	DataBase	Global	0	2	atomIndexPtr	DesignExtractorImpl	-	
2	Copy	Term	Address	DataBase	Imported	0	1	ToolDriver	-	-	
2	FreeString	Term	formSW	DataBase	Public	1	0	-	-	-	
2	MakesWs	nonT	fileSW	DataBase	Public	1	0	-	-	-	

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE				DATA				CONNECTION			
Lvl	Node Name	Type		Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
2	log	nonT		addresses	DataBase	Local	1	0	-	-	-
				addresses	IntCallP	Local	0	1	ToolDriver	MakeSWs	addresses
3	Delete Exit	Term Term		fileSW	IntCallP	Public	0	1	Put	log	fileSW
				s	IntCallP	Imported	0	1	Put	log	-
3	SetPriority	Term		running	DataBase	Global	1	0	running	DesignExtractorImpl	-
				Yield							
3	log	nonT		fileSW	IntCallP	Public	0	1	Put	log	fileSW
				s	IntCallP	Imported	0	1	Put	log	-
3	CheckSelection	nonT		rawData	DataBase	Public	0	2	-	-	-
				SCGraph	DataBase	Public	0	2	-	-	-
3				SCTable	DataBase	Public	0	2	-	-	-
				DFDVariable	DataBase	Public	0	2	-	-	-
3				DFDTable	DataBase	Public	0	2	-	-	-
				ok	DataBase	Local	1	0	-	-	-
3	CheckForAbort	nonT		ok	RetVal	Local	0	1	log	Exit	ok
				window	IntCallP	Public	0	1	UserInput	CheckForAbort	window
3	Control	nonT		created	FmlParam	-	0	0	-	-	-
				aStackPtr	FmlParam	-	0	0	-	-	-
3				aIndexPtr	FmlParam	-	0	0	-	-	-
				extStackPtr	FmlParam	-	0	0	-	-	-
3				maxExternal	FmlParam	-	0	0	-	-	-
				fileStackPtr	FmlParam	-	0	0	-	-	-
3				MaxFiles	FmlParam	-	0	0	-	-	-
				rawData	FmlParam	-	0	0	-	-	-
3				SCGraph	FmlParam	-	0	0	-	-	-
				SCTable	FmlParam	-	0	0	-	-	-
3				DFDVariable	FmlParam	-	0	0	-	-	-
				DFDTable	FmlParam	-	0	0	-	-	-
3				debugg	FmlParam	-	0	0	-	-	-
				repeats	FmlParam	-	0	0	-	-	-
3				created	FmlParam	-	0	0	-	-	-
				rawData	FmlParam	-	0	14	-	-	-
3				result	DataBase	UseFmlCal	0	1	-	-	-
				created	DataBase	Local	2	0	-	-	-
3				aStackPtr	IntCallP	UseFmlCal	0	1	Display	Control	created
				aIndexPtr	IntCallP	UseFmlCal	0	1	Display	Control	aStackPtr
3				fileStackPtr	IntCallP	UseFmlCal	0	1	Display	Control	aIndexPtr
				MaxFiles	IntCallP	UseFmlCal	0	1	Display	Control	fileStackPtr
3				debugg	IntCallP	UseFmlCal	0	1	Display	Control	MaxFiles
				SCGraph	DataBase	UseFmlCal	0	1	-	-	debugg
3				SCTable	DataBase	UseFmlCal	0	1	-	-	-
				DFDVariable	DataBase	UseFmlCal	0	1	-	-	-
3				DFDTable	DataBase	UseFmlCal	0	1	-	-	-
					DataBase	UseFmlCal	0	1	-	-	-

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION					
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias		
3	FreeAtomStack	nonT	created	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	created		
			aStackPtr	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	aStackPtr		
			aIndexPtr	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	aIndexPtr		
			extStackPtr	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	extSP		
			maxExternal	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	maxExt		
			fileStackPtr	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	fileStackPtr		
			MaxFiles	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	MaxFiles		
			SCGraph	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	SCGraph		
			SCTable	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	SCTable		
			DFDVariable	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	DFDVariable		
			DFDTable	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	DFDTable		
			debugg	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	debugg		
			repeats	IntCallP	UsefmlCal	0	1	StructChart	EraseQueue	repeats		
result	RetValue	Local	0	1	StructChart	EraseQueue	result					
aSP	FmlParam	-	0	0	-	-	-					
sh	FmlParam	-	0	0	-	-	-					
aSP	DataBase	UsefmlCal	1	1	-	-	-					
sh	IntCallP	UsefmlCal	0	1	MSegment		FreeAtomStack	sh				
sh	IntCallP	UsefmlCal	1	1	MSegment		FreeAtomStack	sh				
3	AllocateAtomStack	nonT	elements	FmlParam	-	0	0	-	-	-		
			AtomStack	DataBase	Imported	0	1	Parser	-	-	-	
			elements	DataBase	UsefmlCal	0	2	-	-	-	-	
			AtomProps	DataBase	Imported	0	1	Parser	-	-	-	
			wordsPerPage	DataBase	Imported	0	2	Environment	-	-	-	
			sh	DataBase	UsefmlRet	1	0	-	-	-	-	
			aSP	DataBase	UsefmlRet	1	0	-	-	-	-	
			sh	IntCallP	UsefmlRet	0	1	MSegment		AllocateAtomStack	sh	
			length	DataBase	Local	1	1	-	-	-	-	
			sh	IntCallP	UsefmlRet	0	1	MSegment		AllocateAtomStack	sh	
			ClearMetFile	Term								
			Create	Term								
			OpenMetFile	Term								
MetFileLength	Term											
CloseMetFile	Term											
FreeStringHandle	Term											
MaybeQuoted	Term											
Empty	Term											
FreeTokenString	Term											
AppendExtensionIfNe	nonT											
to	FmlParam	-	0	0	-	-	-	-	-			
z	FmlParam	-	0	0	-	-	-	-	-			
length	DataBase	Imported	0	1)	-	-	-	-			
toSS	IntCallP	Local	0	1	String		AppendExtensionIfNe	toSS				
extSS	IntCallP	Local	0	1	String		AppendExtensionIfNe	extSS				
length	IntCallP	Local	0	1	to		-	length				
minLength	IntCallP	Local	0	1	to		-	minLength				
EnumerateOirectory	Term											
ProcessFile	nonT											
LongNumber	Term											

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE				DATA				CONNECTION		
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
3	ForceOutput	Term								
3	Add	Term								
3	delete	Term								
3	CardToString	Term								
3	Activate	Term								
3	Enter	Term	ok	RetVal	UseFmlRet	0	1	-	InputLoc	ok
			running	RetVal	Global	0	1	-	InputLoc	running
			running	RetVal	Global	1	0	-	InputLoc	running
3	Text	Term								
3	OutputProc	Term								
3	Destroy	Term								
3	RemoveCommand	Term								
3	MakeFormSW	Term								
3	MakeForm	nonT	freeDesc	Database	Imported	1	0		-	-
			items	Database	Imported	1	10		-	-
			numItems	IntCallP	Local	0	1	FormSW	MakeForm	numItems
			line0	IntCallP	Imported	0	1	FormSW	MakeForm	FormSW
			string	Database	Imported	0	1		-	-
			inHeap	Database	Imported	0	1		-	-
			sameLine	IntCallP	Imported	0	1	FormSW	MakeForm	FormSW
			switch	Database	Imported	0	6		-	-
			SCGraph	Database	Public	0	1		-	-
			drawBox	Database	Imported	0	6		-	-
			sameLine	IntCallP	Imported	0	1	FormSW	MakeForm	FormSW
			DFDVariable	Database	Public	0	1		-	-
			sameLine	IntCallP	Imported	0	1	FormSW	MakeForm	FormSW
			rawData	Database	Imported	0	1	FormSW	MakeForm	FormSW
			line1	IntCallP	Public	0	1		-	-
			sameLine	IntCallP	Imported	0	1	FormSW	MakeForm	FormSW
			sameLine	IntCallP	Imported	0	1	FormSW	MakeForm	FormSW
			sameLine	IntCallP	Imported	0	1	FormSW	MakeForm	FormSW
			SCTable	Database	Imported	0	1		-	-
			sameLine	IntCallP	Public	0	1	FormSW	MakeForm	FormSW
			DFDTable	Database	Imported	0	1		-	-
			sameLine	IntCallP	Public	0	1	FormSW	MakeForm	FormSW
			repeats	Database	Imported	0	1	FormSW	MakeForm	FormSW
			items	RetVal	Imported	0	1	FormSW	MakeForm	FormSW
			freeDesc	RetVal	Imported	0	1	FormSW	MakeForm	FormSW
3	UnusedLogName	Term								
3	MakeFileSW	Term								
3	NoteSWs	Term								
4	Text	Term								
4	log	nonT	fileSW	IntCallP	Public	0	1	Put	log	fileSW
			s	IntCallP	Imported	0	1	Put	log	-
4	UserAbort	Term								
4	RawData	Term								
4	StructChart	nonT	created	FmlParam	-	0	0	-	-	-

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
	aStackPtr			FmlParam	-	0	0	-		-
	aIndexPtr			FmlParam	-	0	0	-		-
	extSP			FmlParam	-	0	0	-		-
	maxExt			FmlParam	-	0	0	-		-
	fileStackPtr			FmlParam	-	0	0	-		-
	MaxFiles			FmlParam	-	0	0	-		-
	SCGraph			FmlParam	-	0	0	-		-
	SCTable			FmlParam	-	0	0	-		-
	DFDVariable			FmlParam	-	0	0	-		-
	DFDTable			FmlParam	-	0	0	-		-
	debugg			FmlParam	-	0	0	-		-
	repeats			FmlParam	-	0	0	-		-
	created			FmlParam	-	0	13	-		-
	posIndex			DataBase	Global	1	0	posIndex	StructChartImp)	-
	fullCount			DataBase	Global	1	0	fullCount	StructChartImp)	-
	positionStackPtr			DataBase	Global	1	0	positionStackPtr	StructChartImp)	-
	PositionStack			DataBase	Global	0	1	PositionStack	StructChartImp)	-
	fullStackPtr			DataBase	Global	1	0	fullStackPtr	StructChartImp)	-
	FullStack			DataBase	Imported	0	1	Display	-	-
	aIndexPtr			DataBase	UsefmlCal	0	1	-	StructChartImp)	-
	statsFile			DataBase	External	1	1	statsFile	-	-
	text			DataBase	Imported	0	1	-	StructChartImp)	-
	statsFile			IntCallP	External	1	1	statsFile	StructChart	statsFile
	aIndexPtr			IntCallP	UsefmlCal	0	1	FormatData	StructChart	aIP
	aStackPtr			IntCallP	UsefmlCal	0	1	FormatData	StructChart	atomStackPtr
	fileStackPtr			IntCallP	UsefmlCal	0	1	FormatData	StructChart	fileStackPtr
	MaxFiles			IntCallP	UsefmlCal	0	1	FormatData	StructChart	MaxFiles
	SCGraph			IntCallP	UsefmlCal	0	1	FormatData	StructChart	SCGraph
	SCTable			IntCallP	UsefmlCal	0	1	FormatData	StructChart	SCTable
	debugg			IntCallP	UsefmlCal	0	1	FormatData	StructChart	debugg
	repeats			IntCallP	UsefmlCal	0	1	FormatData	StructChart	repeats
	SCGraph			DataBase	UsefmlCal	0	1	-	-	-
	SCTable			DataBase	UsefmlCal	0	1	-	-	-
	aIndexPtr			IntCallP	UsefmlCal	0	1	Display	StructChart	aIndexPtr
	aStackPtr			IntCallP	UsefmlCal	0	1	Display	StructChart	aStackPtr
	extSP			IntCallP	UsefmlCal	0	1	Display	StructChart	extSP
	maxExt			IntCallP	UsefmlCal	0	1	Display	StructChart	maxExt
	fileStackPtr			IntCallP	UsefmlCal	0	1	Display	StructChart	fileStackPtr
	MaxFiles			IntCallP	UsefmlCal	0	1	Display	StructChart	MaxFiles
	positionStackPtr			IntCallP	Global	0	2	Display	StructChart	positionStackPtr
	fullStackPtr			IntCallP	Global	0	2	Display	StructChart	fullStackPtr
	fullCount			IntCallP	Global	0	1	Display	StructChart	fullCount
	DFDVariable			IntCallP	UsefmlCal	0	1	Display	StructChart	DFDVariable
	DFDTable			IntCallP	UsefmlCal	0	1	Display	StructChart	DFDTable
	debugg			IntCallP	UsefmlCal	0	1	Display	StructChart	debugg
	statsFile			IntCallP	External	0	1	MStream	StructChart	statsFile
	statsFile			IntCallP	External	0	1	Stream	StructChart	statsFile
	statsFile			IntCallP	External	1	1	statsFile	StructChart	statsFile
	result			DataBase	Local	1	0	-	-	-
	result			RetVal	Local	0	1	log	Exit	result

4

Address

Term

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA						CONNECTION		
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias	
4	Delete	Term	ok	FmlParam	-	0	0	-	-	-	
4	Create	Term	file	DataBase	External	0	1	file	ProcessFile	-	
4	EquivalentSubString	Term	delete	DataBase	Imported	0	1	file	-	-	
4	to	Term	file	IntCallP	External	2	1	file	CleanUp	file	
4	AppendStringAndGrow	Term	ok	IntCallP	UseFmlCal	0	1	log	Exit	ok	
4	CleanUp	nonT	warnings	IntCallP	Local	1	3	log	Exit	warnings	
			log	IntCallP	Local	0	2	log	Exit	-	
			CardToSting	DataBase	Imported	0	2	Utils	-	-	
			fileStackPtr	DataBase	Imported	0	2	fileStackPtr	DesignExtractorImpl	-	
			fileCount	DataBase	Global	0	7	fileCount	DesignExtractorImpl	-	
			endIndex	DataBase	Global	0	1	fileStackPtr	CleanUp	-	
			fileCount	IntCallP	Dereferen	0	1	fileStackPtr	CleanUp	fileCount	
			atomStackPtr	IntCallP	Global	1	10	Utils	-	-	
			atomIndexPtr	DataBase	Global	0	2	atomStackPtr	DesignExtractorImpl	-	
			sCount	DataBase	Global	0	2	atomIndexPtr	DesignExtractorImpl	-	
			totalErrors	DataBase	Dereferen	0	1	atomStackPtr	CleanUp	-	
			totalWarnings	DataBase	Dereferen	1	1	totalErrors	DesignExtractorImpl	-	
				DataBase	Global	1	1	totalWarnings	DesignExtractorImpl	-	
4	CheckForAbort	nonT	window	IntCallP	Public	0	1	UserInput	CheckForAbort	window	
4	Line	Term									
4	CopyToNewString	Term									
4	CardToSting	Term									
4	FreeString	Term									
4	SetPosition	Term									
4	ScanInit	Term									
4	Parse	nonT	cz	FmlParam	-	0	0	-	-	-	
			aStackPtr	FmlParam	-	0	0	-	-	-	
			aIndexPtr	FmlParam	-	0	0	-	-	-	
			fileStackPtr	FmlParam	-	0	0	-	-	-	
			fileCount	FmlParam	-	0	0	-	-	-	
			rowData	FmlParam	-	0	0	-	-	-	
			cz	FmlParam	-	0	0	-	-	-	
			z	FmlParam	-	0	7	-	-	-	
			cz	DataBase	Global	1	0	z	ParserImpl	-	
			input	DataBase	UseFmlCal	0	1	-	-	-	
			Atom	DataBase	Imported	1	0	-	-	-	
			nErrors	DataBase	Imported	0	1	Parser	-	-	
			complete	DataBase	Local	1	0	-	-	-	
			i	DataBase	Local	1	0	-	-	-	
			top	DataBase	Local	5	11	-	-	-	
			valid	DataBase	Global	1	1	top	ParserImpl	-	
			qi	DataBase	Local	3	6	-	-	-	
			s	DataBase	Global	1	0	qi	ParserImpl	-	
				DataBase	Global	0	6	s	ParserImpl	-	

DATA FLDW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA					CONNECTION		
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
	currentState		currentState	DataBase	Local	4	4	-	-	-
	InitialState		InitialState	DataBase	Imported	0	1	-	-	-
	class		class	DataBase	Imported	3	0	lastToken	-	-
	NullSymbol		NullSymbol	DataBase	Global	0	2	NullSymbol	ParserImpl	-
	inputSymbol		inputSymbol	DataBase	Global	1	2	inputSymbol	ParserImpl	-
	InitialSymbol		InitialSymbol	DataBase	Imported	0	1	-	-	-
	inputValue		inputValue	DataBase	Global	1	0	inputValue	ParserImpl	-
	z		z	IntCallp	Global	0	1	String	Parse	z
	inputLoc		inputLoc	DataBase	Global	1	3	inputLoc	ParserImpl	-
	ids		ids	DataBase	Public	0	1	-	-	-
	z		z	IntCallp	Global	0	1	String	Parse	z
	idindex		idindex	DataBase	Public	1	0	-	-	-
	FinalState		FinalState	DataBase	Imported	0	1	-	-	-
	tokenID		tokenID	DataBase	Imported	0	1	-	-	-
	inputValue		inputValue	IntCallp	Global	0	1	Utils	Parse	inputValue
	ids		ids	IntCallp	Public	0	2	Utils	Parse	ids
	idindex		idindex	IntCallp	Public	2	4	Utils	Parse	idindex
	maxIds		maxIds	DataBase	Imported	0	1	Parser	-	-
	inputValue		inputValue	IntCallp	Global	0	1	String	Parse	inputValue
	z		z	IntCallp	Global	0	1	String	Parse	z
	tI		tI	DataBase	Local	0	5	-	-	-
	tLength		tLength	DataBase	Global	0	1	tLength	ParserImpl	-
	tSymbol		tSymbol	DataBase	Global	0	1	tSymbol	ParserImpl	-
	DefaultMarker		DefaultMarker	DataBase	Imported	0	1	-	-	-
	SyntaxError		SyntaxError	DataBase	Imported	0	2	-	-	-
	action		action	DataBase	Local	3	1	-	-	-
	tAction		tAction	DataBase	Global	0	1	tAction	ParserImpl	-
	k		k	DataBase	Local	0	2	-	-	-
	ruleValue		ruleValue	DataBase	Global	0	1	ruleValue	ParserImpl	-
	qI		qI	IntCallp	Local	3	4	Parser	Parse	qI
	top		top	IntCallp	Global	2	7	Parser	Parse	top
	aStackPtr		aStackPtr	DataBase	UsefmlCal	0	1	-	-	-
	unique		unique	DataBase	Local	0	2	-	-	-
	inputValue		inputValue	IntCallp	Global	0	1	Parser	Parse	inputValue
	inputSymbol		inputSymbol	IntCallp	Global	0	4	Parser	Parse	inputSymbol
	aIndexPtr		aIndexPtr	IntCallp	UsefmlCal	0	6	Parser	Parse	aIndexPtr
	aStackPtr		aStackPtr	IntCallp	UsefmlCal	0	1	Parser	Parse	aStackPtr
	fileCount		fileCount	IntCallp	UsefmlCal	0	1	Parser	Parse	fileCount
	ruleValue		ruleValue	IntCallp	Global	0	1	Parser	Parse	ruleValue
	inputValue		inputValue	IntCallp	Global	0	4	String	Parse	inputValue
	MaxString		MaxString	DataBase	Imported	0	1	Parser	-	-
	length		length	DataBase	Imported	0	3	s	-	-
	v		v	DataBase	Global	0	2	v	ParserImpl	-
	l		l	DataBase	Global	0	2	l	ParserImpl	-
	tag		tag	DataBase	Imported	0	1	action	-	-
	Scan		Scan	DataBase	Global	0	1	Scan	ParserImpl	-
	q		q	DataBase	Global	0	1	q	ParserImpl	-
	pLength		pLength	DataBase	Imported	0	1	action	-	-
	lhs		lhs	DataBase	Dereferen	0	1	s	Parse	-
	NTState		NTState	DataBase	Imported	0	1	-	-	-
	nI		nI	DataBase	Local	0	5	-	-	-

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE				DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias	
	nLength		nLength	DataBase	Global	0	1	nLength	ParserImpl	-	
	lhs		lhs	DataBase	Local	0	2	-	-	-	
	nSymbol		nSymbol	DataBase	Global	0	1	nSymbol	ParserImpl	-	
	nAction		nAction	DataBase	Global	0	1	nAction	ParserImpl	-	
	nFound		nFound	DataBase	Imported	0	2	-	-	-	
	ntDefaults		ntDefaults	DataBase	Global	0	1	ntDefaults	ParserImpl	-	
	m		m	DataBase	Local	1	1	-	-	-	
	transition		transition	DataBase	Imported	0	1	action	-	-	
	value		value	DataBase	Imported	1	0	lastToken	-	-	
	index		index	DataBase	Imported	1	0	lastToken	-	-	
	log		log	IntCallip	Imported	0	1	Format	Parse	Utils	
	index		index	IntCallip	Imported	0	1	Format	Parse	lastToken	
	inputValue		inputValue	IntCallip	Global	0	1	Parser	Parse	inputValue	
	inputSymbol		inputSymbol	IntCallip	Global	0	1	Parser	Parse	inputSymbol	
	aIndexPtr		aIndexPtr	IntCallip	UseFmlCal	0	1	Parser	Parse	aIndexPtr	
	aStackPtr		aStackPtr	IntCallip	UseFmlCal	0	1	Parser	Parse	aStackPtr	
	fileStackPtr		fileStackPtr	IntCallip	UseFmlCal	0	1	Parser	Parse	fileStackPtr	
	fileCount		fileCount	IntCallip	UseFmlCal	0	1	Parser	Parse	fileCount	
	ruleValue		ruleValue	IntCallip	Global	0	1	Parser	Parse	ruleValue	
	inputValue		inputValue	IntCallip	Global	0	1	Parser	Parse	inputValue	
	inputSymbol		inputSymbol	IntCallip	Global	0	1	Parser	Parse	inputSymbol	
	aIndexPtr		aIndexPtr	IntCallip	UseFmlCal	0	1	Parser	Parse	aIndexPtr	
	aStackPtr		aStackPtr	IntCallip	UseFmlCal	0	1	Parser	Parse	aStackPtr	
	fileStackPtr		fileStackPtr	IntCallip	UseFmlCal	0	1	Parser	Parse	fileStackPtr	
	fileCount		fileCount	IntCallip	UseFmlCal	0	1	Parser	Parse	fileCount	
	ruleValue		ruleValue	IntCallip	Global	0	1	Parser	Parse	ruleValue	
	aIndexPtr		aIndexPtr	IntCallip	UseFmlCal	0	1	Parser	Parse	aIndexPtr	
	aStackPtr		aStackPtr	IntCallip	UseFmlCal	0	1	Parser	Parse	aStackPtr	
	fileStackPtr		fileStackPtr	IntCallip	UseFmlCal	0	1	Parser	Parse	fileStackPtr	
	fileCount		fileCount	IntCallip	UseFmlCal	0	1	Parser	Parse	fileCount	
	aStackPtr		aStackPtr	RetValue	UseFmlCal	0	1	ParseReset	ParseInit	aStackPtr	
4	AllocateItemDescrip	Term									
4	StringItem	Term									
4	BooleanItem	Term									
4	CommandItem	Term									
5	Text	Term									
5	CheckForAbort	nonT	window	IntCallip	Public	0	1	UserInput	CheckForAbort	window	
5	WriteOnly	Term									
5	delete	Term									
5	FormatData	nonT									
	aIP		aIP	FmlParam	-	0	0	-	-	-	
	atomStackPtr		atomStackPtr	FmlParam	-	0	0	-	-	-	
	fileStackPtr		fileStackPtr	FmlParam	-	0	0	-	-	-	
	MaxFiles		MaxFiles	FmlParam	-	0	0	-	-	-	
	SCGraph		SCGraph	FmlParam	-	0	0	-	-	-	
	SCTable		SCTable	FmlParam	-	0	0	-	-	-	

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
	debugg		debugg	FmlParam	-	0	0	-	-	-
	repeats		repeats	FmlParam	-	0	0	-	-	-
	stackIndex		stackIndex	DataBase	Global	4	5	stackIndex	StructChartImpl	-
	lineCount		lineCount	DataBase	Global	1	0	lineCount	StructChartImpl	-
	posIndex		posIndex	DataBase	Global	4	4	posIndex	StructChartImpl	-
	SCGraph		SCGraph	DataBase	UsefmlCal	0	2	-	-	-
	SCTable		SCTable	DataBase	UsefmlCal	0	2	-	-	-
	fileStackPtr		fileStackPtr	IntCallP	UsefmlCal	0	1	WriteFiles	StudyRepeats	fileStackPtr
	MaxFiles		MaxFiles	IntCallP	UsefmlCal	0	1	WriteFiles	StudyRepeats	MaxFiles
	aIP		aIP	IntCallP	UsefmlCal	0	1	WriteExportInfo	WriteFiles	aIP
	atomStackPtr		atomStackPtr	IntCallP	UsefmlCal	0	1	WriteExportInfo	WriteFiles	aSP
	fileStackPtr		fileStackPtr	IntCallP	UsefmlCal	0	1	WriteExportInfo	WriteFiles	fSP
	MaxFiles		MaxFiles	IntCallP	UsefmlCal	0	1	WriteExportInfo	WriteFiles	MaxFiles
	fileStackPtr		fileStackPtr	DataBase	UsefmlCal	0	1	-	-	-
	endIndex		endIndex	DataBase	Dereferen	0	1	fileStackPtr	FormatData	-
	atomStackPtr		atomStackPtr	IntCallP	UsefmlCal	0	1	CheckReturnConditio	SearchForDeclaratio	aSP
	debugg		debugg	IntCallP	UsefmlCal	0	1	CheckReturnConditio	SearchForDeclaratio	debugg
	atomStackPtr		atomStackPtr	DataBase	UsefmlCal	0	1	-	-	-
	idMark		idMark	DataBase	Oereferen	0	1	atomStackPtr	FormatData	-
	atomStackPtr		atomStackPtr	IntCallP	UsefmlCal	0	1	CheckIfTerminalDecl	CleanUpPositionData	aSP
	fileStackPtr		fileStackPtr	IntCallP	UsefmlCal	0	1	CheckIfTerminalDecl	CleanUpPositionData	fSP
	SLC		SLC	DataBase	Local	4	0	-	-	-
	atomStackPtr		atomStackPtr	IntCallP	UsefmlCal	0	1	CollectPositionData	CheckIfTerminalDecl	aSP
	debugg		debugg	IntCallP	UsefmlCal	0	1	CollectPositionData	CheckIfTerminalDecl	debugg
	atomStackPtr		atomStackPtr	IntCallP	UsefmlCal	0	1	UpdateFullStack	SemiGraphOutput	aSP
	fileStackPtr		fileStackPtr	IntCallP	UsefmlCal	0	1	UpdateFullStack	SemiGraphOutput	fSP
	SLC		SLC	IntCallP	Local	0	1	UpdateFullStack	SemiGraphOutput	found
	found		found	IntCallP	Local	0	1	UpdateFullStack	SemiGraphOutput	aSP
	atomStackPtr		atomStackPtr	IntCallP	UsefmlCal	0	1	SuppressOutput	GoFindCallParams	debugg
	debugg		debugg	IntCallP	UsefmlCal	0	1	SuppressOutput	GoFindCallParams	aSP
	atomStackPtr		atomStackPtr	IntCallP	UsefmlCal	0	1	SuppressOutput	GoFindCallParams	debugg
	debugg		debugg	IntCallP	UsefmlCal	0	1	CollectPositionData	CheckIfTerminalDecl	aSP
	atomStackPtr		atomStackPtr	IntCallP	UsefmlCal	0	1	CollectPositionData	CheckIfTerminalDecl	debugg
	fileStackPtr		fileStackPtr	IntCallP	UsefmlCal	0	1	UpdateFullStack	SemiGraphOutput	aSP
	SLC		SLC	IntCallP	UsefmlCal	0	1	UpdateFullStack	SemiGraphOutput	fSP
	found		found	IntCallP	Local	0	1	UpdateFullStack	SemiGraphOutput	SLC
	found		found	IntCallP	Local	0	1	UpdateFullStack	SemiGraphOutput	found
	atomStackPtr		atomStackPtr	DataBase	Local	1	2	-	-	-
	fileStackPtr		fileStackPtr	IntCallP	UsefmlCal	0	1	SearchForDeclaratio	SuppressOutput	aSP
	MaxFiles		MaxFiles	IntCallP	UsefmlCal	0	1	SearchForDeclaratio	SuppressOutput	fSP
	atomStackPtr		atomStackPtr	IntCallP	UsefmlCal	0	1	SearchForDeclaratio	SuppressOutput	MaxFiles
	debugg		debugg	IntCallP	UsefmlCal	0	1	CollectPositionData	CheckIfTerminalDecl	aSP
	atomStackPtr		atomStackPtr	IntCallP	UsefmlCal	0	1	CollectPositionData	CheckIfTerminalDecl	debugg
	fileStackPtr		fileStackPtr	IntCallP	UsefmlCal	0	1	UpdateFullStack	SemiGraphOutput	aSP
	SLC		SLC	IntCallP	UsefmlCal	0	1	UpdateFullStack	SemiGraphOutput	fSP
	found		found	IntCallP	Local	0	1	UpdateFullStack	SemiGraphOutput	SLC
	found		found	IntCallP	Local	0	1	UpdateFullStack	SemiGraphOutput	found
	aIP		aIP	IntCallP	Local	0	1	GoFindCallParams	UpdateFullStack	aIP
	atomStackPtr		atomStackPtr	IntCallP	UsefmlCal	0	1	GoFindCallParams	UpdateFullStack	found
	fullStackPtr		fullStackPtr	IntCallP	UsefmlCal	0	1	GoFindCallParams	UpdateFullStack	aIP
	fullStackPtr		fullStackPtr	IntCallP	Global	0	1	GoFindCallParams	UpdateFullStack	fullStackPtr

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
5	log	nonT	aIP	DataBase	UseFmlCal	0	1	-	-	-
			atomStackPtr	IntCallP	UseFmlCal	0	1	StudyRepeats	FindFileNumber	aSP
			atomStackPtr	IntCallP	UseFmlCal	0	1	SemiGraphOutput	CollectPositionData	aSP
			fileStackPtr	IntCallP	UseFmlCal	0	1	SemiGraphOutput	CollectPositionData	fSP
			debugg	IntCallP	UseFmlCal	0	1	SemiGraphOutput	CollectPositionData	debugg
			repeats	IntCallP	UseFmlCal	0	1	SemiGraphOutput	CollectPositionData	repeats
			atomStackPtr	IntCallP	UseFmlCal	0	1	TabularOutput	CheckProcExists	aSP
			fileStackPtr	IntCallP	UseFmlCal	0	1	TabularOutput	CheckProcExists	fSP
			debugg	IntCallP	UseFmlCal	0	1	TabularOutput	CheckProcExists	debugg
			repeats	IntCallP	UseFmlCal	0	1	TabularOutput	CheckProcExists	repeats
5	log	nonT	fileSW	IntCallP	Public	0	1	Put	log	fileSW
	s		IntCallP	Imported	0	1	Put	log	-	
	DataFlow	Term								
	SetLength	Term								
	GetPosition	Term								
	CardIoString	Term								
	UserAbort	Term								
	ParseInit	nonT	tablePtr	DataBase	Global	1	10	tablePtr	ParserImpl	-
			MesaTab	IntCallP	Global	0	1	Runtime	ParseInit	MesaTab
			tStart	DataBase	Global	1	1	tStart	ParserImpl	-
5	ParseReset	nonT	tLength	DataBase	Global	1	1	tLength	ParserImpl	-
	CopyToNewString	Term	tSymbol	DataBase	Global	1	1	tSymbol	ParserImpl	-
	StringEqual	Term	tAction	DataBase	Global	1	1	tAction	ParserImpl	-
	ProcessQueue	Term	nStart	DataBase	Global	1	1	nStart	ParserImpl	-
	BaseTable	Term	nLength	DataBase	Global	1	1	nLength	ParserImpl	-
	Equal	Term	nSymbol	DataBase	Global	1	1	nSymbol	ParserImpl	-
	ExpandStack	nonT	nAction	DataBase	Global	1	1	nAction	ParserImpl	-
			ntDefaults	DataBase	Global	1	1	ntDefaults	ParserImpl	-
			prodData	DataBase	Global	1	1	prodData	ParserImpl	-
			s	DataBase	Global	1	0	s	ParserImpl	-
		q	DataBase	Global	1	0	q	ParserImpl	-	
5	input	Term	delta	FmlParam	-	0	0	-	-	-
	ExpandQueue	nonT	s	DataBase	Global	1	2	s	ParserImpl	-
			length	DataBase	Imported	0	1	s	-	-
			oldSize	DataBase	Local	0	2	-	-	-
			delta	DataBase	UseFmlCal	0	1	-	-	-
			i	DataBase	Local	0	6	-	-	-
			l	DataBase	Global	1	1	l	ParserImpl	-
			v	DataBase	Global	1	1	v	ParserImpl	-
			delta	FmlParam	-	0	0	-	-	-

OATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			OATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
6	log	nonT	z	IntCallip	Global	0	2	String	WriteFiles	z
			numString	IntCallip	Global	1	1	String	WriteFiles	numString
			endIndex	DataBase	Dereferen	0	1	fileStackPtr	WriteFiles	-
			maxEndLength	IntCallip	Local	0	1	CheckLength	StatsWrite	length
			numString	IntCallip	Global	0	1	StatsWrite	WriteSCTitles	str
			z	IntCallip	Global	0	2	String	WriteFiles	z
			numString	IntCallip	Global	1	1	String	WriteFiles	numString
			exportName	DataBase	Dereferen	0	1	fileStackPtr	WriteFiles	-
			maxExportNameLength	IntCallip	Local	0	1	CheckLength	StatsWrite	length
			numString	IntCallip	Global	0	1	StatsWrite	WriteSCTitles	str
			z	IntCallip	Global	0	1	String	WriteFiles	z
			numString	IntCallip	Global	0	1	String	WriteFiles	numString
			statsFile	IntCallip	External	0	1	Stream	WriteFiles	statsFile
			CR	IntCallip	Imported	0	1	Stream	WriteFiles	Ascii
6	WriteExportInfo	nonT	fileSW	IntCallip	Public	0	1	Put	log	fileSW
			s	IntCallip	Imported	0	1	Put	log	-
			aIP	FmlParam	-	0	0	-	-	-
			aSP	FmlParam	-	0	0	-	-	-
			fSP	FmlParam	-	0	0	-	-	-
			MaxFiles	FmlParam	-	0	0	-	-	-
			exportStackPtr	DataBase	Global	1	4	exportStackPtr	StructChartImpl	-
			ExportStack	DataBase	Global	0	1	ExportStack	StructChartImpl	-
			tempIndex	DataBase	Local	1	7	-	-	-
			aIP	DataBase	UsefmlCal	0	1	-	-	-
			aSP	DataBase	UsefmlCal	0	7	-	-	-
			idMark	DataBase	Dereferen	0	1	aSP	WriteExportInfo	-
			procCall	DataBase	Local	0	1	-	-	-
			Equal	DataBase	Imported	0	3	String	-	-
IEatom	DataBase	Dereferen	0	1	aSP	WriteExportInfo	-			
			tempFileCount	DataBase	Local	2	4	-	-	-
			MaxFiles	DataBase	UsefmlCal	0	1	-	-	-
			fSP	DataBase	UsefmlCal	0	2	-	-	-
			fileCount	DataBase	Dereferen	0	1	fSP	WriteExportInfo	-
			IEatom	DataBase	Dereferen	0	1	aSP	WriteExportInfo	-
			exportName	DataBase	Dereferen	0	1	fSP	WriteExportInfo	-
			numString	DataBase	Global	1	0	numString	StructChartImpl	-
			CopyToNewString	DataBase	Imported	0	1	String	-	-
			inputString	DataBase	Dereferen	0	1	aSP	WriteExportInfo	-
			z	DataBase	Global	0	1	z	StructChartImpl	-
			OK	DataBase	Local	1	3	-	-	-
			aSP	IntCallip	UsefmlCal	0	1	CheckProcExists	ReduceLength	aSP
			fSP	IntCallip	UsefmlCal	0	1	CheckProcExists	ReduceLength	fSP
			tempFileCount	IntCallip	Local	0	1	CheckProcExists	ReduceLength	tempFileCount
numString	IntCallip	Global	0	1	CheckProcExists	ReduceLength	numString			
z	IntCallip	Global	0	1	String	WriteExportInfo	z			
numString	IntCallip	Global	0	1	String	WriteExportInfo	numString			

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
6	CheckForAbort	nonT	exportStackCount	DataBase	Local	2	6	-	-	-
			missingProcLocation	DataBase	Dereferen	1	0	exportStackPtr	WriteExportInfo	-
			Index	DataBase	Dereferen	0	1	aSP	WriteExportInfo	-
			loopCount	DataBase	Local	2	3	-	-	-
			currentCall	DataBase	Local	1	1	-	-	-
			missingProcLocation	DataBase	Dereferen	0	1	exportStackPtr	WriteExportInfo	-
			lastCall	DataBase	Local	1	1	-	-	-
			missingProcLocation	DataBase	Dereferen	0	1	exportStackPtr	WriteExportInfo	-
			inputString	DataBase	Dereferen	0	1	aSP	WriteExportInfo	-
			inputString	DataBase	Dereferen	0	1	aSP	WriteExportInfo	-
			exportStackCount	IntCallP	Local	0	1	OutputExportInfo	WriteExportInfo	eSC
			exportStackPtr	IntCallP	Global	0	1	OutputExportInfo	WriteExportInfo	eSP
			fSP	IntCallP	UsefmlCal	0	1	OutputExportInfo	WriteExportInfo	fSP
			aSP	IntCallP	UsefmlCal	0	1	OutputExportInfo	WriteExportInfo	aSP
			window	IntCallP	Public	0	1	UserInput	CheckForAbort	window
6	CheckReturnConditio	nonT	aSP	FmlParam	-	0	0	-	-	-
			debugg	FmlParam	-	0	0	-	-	-
			positionStackPtr	DataBase	Global	0	7	positionStackPtr	StructChartImpl	-
			stackIndex	DataBase	Global	3	11	stackIndex	StructChartImpl	-
			procLevel	DataBase	Dereferen	0	1	positionStackPtr	CheckReturnConditio	-
			callLocation	DataBase	Dereferen	0	1	positionStackPtr	CheckReturnConditio	-
			dProcLevel	DataBase	Dereferen	0	1	positionStackPtr	CheckReturnConditio	-
			aSP	DataBase	UsefmlCal	0	3	-	-	-
			posIndex	DataBase	Global	2	2	posIndex	StructChartImpl	-
			plevel	DataBase	Dereferen	0	1	aSP	CheckReturnConditio	-
			callLocation	DataBase	Dereferen	0	1	positionStackPtr	CheckReturnConditio	-
			debugg	DataBase	UsefmlCal	0	3	-	-	-
			numString	DataBase	Global	1	0	numString	StructChartImpl	-
			posIndex	IntCallP	Global	0	2	Utils	CheckReturnConditio	posIndex
			z	DataBase	Global	0	1	z	StructChartImpl	-
			maxIndexLength	IntCallP	Global	0	1	CheckLength	StatsWrite	length
			numString	IntCallP	Global	0	1	StatsWrite	WriteSCTitles	str
			z	IntCallP	Global	0	2	String	CheckReturnConditio	z
			numString	IntCallP	Global	1	1	String	CheckReturnConditio	numString
			statsFile	IntCallP	External	0	1	Stream	CheckReturnConditio	statsFile
			CR	IntCallP	Imported	0	1	Stream	CheckReturnConditio	Ascii
			dProcLevel	DataBase	Dereferen	0	1	positionStackPtr	CheckReturnConditio	-
			plevel	DataBase	Dereferen	0	1	aSP	CheckReturnConditio	-
			posIndex	IntCallP	Global	2	3	Utils	CheckReturnConditio	posIndex
			maxIndexLength	IntCallP	Global	0	1	CheckLength	StatsWrite	length
			numString	IntCallP	Global	0	1	StatsWrite	WriteSCTitles	str
			z	IntCallP	Global	0	2	String	CheckReturnConditio	z
			numString	IntCallP	Global	1	1	String	CheckReturnConditio	numString
			statsFile	IntCallP	External	0	1	Stream	CheckReturnConditio	statsFile
			CR	IntCallP	Imported	0	1	Stream	CheckReturnConditio	Ascii
			dProcLevel	DataBase	Dereferen	0	1	positionStackPtr	CheckReturnConditio	-
			plevel	DataBase	Dereferen	0	1	aSP	CheckReturnConditio	-
			callLocation	DataBase	Dereferen	0	1	positionStackPtr	CheckReturnConditio	-
posIndex	IntCallP	Global	0	1	Utils	CheckReturnConditio	posIndex			

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION				
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias	
6	CheckIfTerminalDecl	Term	maxIndexLength	IntCallip	Global	0	1	CheckLength	StatsWrite	length	
			numString	IntCallip	Global	0	1	StatsWrite	WriteSCTitles	str	
			z	IntCallip	Global	0	1	String	CheckReturnConditio	z	numString
			numString	IntCallip	Global	0	1	String	CheckReturnConditio	CheckReturnConditio	statsFile
			statsFile	IntCallip	External	0	1	Stream	CheckReturnConditio	CheckReturnConditio	Ascii
			CR	IntCallip	Imported	0	1	Stream			
			aSP	FmlParam	-	0	0	-	-	-	-
			fSP	FmlParam	-	0	0	-	-	-	-
			tempIndex	DataBase	Local	1	4	-	-	-	-
			fSP	DataBase	UsefmlCal	0	1	-	-	-	-
			tempFileCount	DataBase	Local	0	1	-	-	-	-
			endIndex	DataBase	Dereferen	0	1	fSP	CheckIfTerminalDecl	CheckIfTerminalDecl	-
			aSP	DataBase	UsefmlCal	0	3	-	-	-	-
			idMark	DataBase	Dereferen	0	1	aSP	CheckIfTerminalDecl	CheckIfTerminalDecl	-
			procCall	DataBase	Local	0	1	-	-	-	-
			Equal	DataBase	Imported	0	1	String	StructChartImpl	StructChartImpl	-
			posIndex	DataBase	Global	0	1	posIndex	CheckIfTerminalDecl	CheckIfTerminalDecl	-
			inputString	DataBase	Dereferen	0	1	aSP	CheckIfTerminalDecl	CheckIfTerminalDecl	-
			inputString	DataBase	Dereferen	0	1	aSP	CheckIfTerminalDecl	CheckIfTerminalDecl	-
			terminal	DataBase	Local	1	0	-	-	-	-
			terminal	RetVal	Local	0	1	TabularOutput	CheckProcExists	terminal	terminal
6	CleanUpPositionData	Term	positionStackPtr	DataBase	Global	0	4	positionStackPtr	StructChartImpl	-	
			stackIndex	DataBase	Global	0	4	stackIndex	StructChartImpl	StructChartImpl	-
			declLocation	DataBase	Dereferen	1	0	positionStackPtr	CleanUpPositionData	CleanUpPositionData	-
			dProcLevel	DataBase	Dereferen	1	0	positionStackPtr	CleanUpPositionData	CleanUpPositionData	-
			callLocation	DataBase	Dereferen	1	0	positionStackPtr	CleanUpPositionData	CleanUpPositionData	-
			procLevel	DataBase	Dereferen	1	0	positionStackPtr	CleanUpPositionData	CleanUpPositionData	-
			aSP	FmlParam	-	0	0	-	-	-	-
			debugg	FmlParam	-	0	0	-	-	-	-
			aSP	DataBase	UsefmlCal	0	3	-	-	-	-
			posIndex	DataBase	Global	0	5	posIndex	StructChartImpl	StructChartImpl	-
6	CollectPositionData	nonT	idMark	DataBase	Dereferen	0	1	aSP	CollectPositionData	CollectPositionData	-
			procedure	DataBase	Local	0	1	-	-	-	-
			positionStackPtr	DataBase	Global	0	8	positionStackPtr	StructChartImpl	StructChartImpl	-
			stackIndex	DataBase	Global	0	8	stackIndex	StructChartImpl	StructChartImpl	-
			declLocation	DataBase	Dereferen	1	0	positionStackPtr	CollectPositionData	CollectPositionData	-
			dProcLevel	DataBase	Dereferen	1	0	positionStackPtr	CollectPositionData	CollectPositionData	-
			plevel	DataBase	Dereferen	0	1	aSP	CollectPositionData	CollectPositionData	-
			callLocation	DataBase	Dereferen	1	0	positionStackPtr	CollectPositionData	CollectPositionData	-
			procLevel	DataBase	Dereferen	1	0	positionStackPtr	CollectPositionData	CollectPositionData	-
			plevel	DataBase	Dereferen	0	1	aSP	CollectPositionData	CollectPositionData	-
			debugg	DataBase	UsefmlCal	0	1	-	-	-	-
			numString	DataBase	Global	1	0	numString	StructChartImpl	StructChartImpl	-
			CopyToNewString	DataBase	Imported	0	4	String	-	-	-
			CardToString	DataBase	Imported	0	4	Utils	-	-	-
			callLocation	DataBase	Dereferen	0	1	positionStackPtr	CollectPositionData	CollectPositionData	-
			z	DataBase	Global	0	1	z	StructChartImpl	StructChartImpl	-
			numString	IntCallip	Global	0	1	StatsWrite	WriteSCTitles	WriteSCTitles	str

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA					CONNECTION		
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
6	UpdateFullStack	nonT	z	IntCallP	Global	0	2	String	CollectPositionData	z
			numString	IntCallP	Global	1	1	String	CollectPositionData	numString
			procLevel	DataBase	Dereferen	0	1	positionStackPtr	CollectPositionData	-
			numString	IntCallP	Global	0	1	StatsWrite	WriteSCTitles	str
			z	IntCallP	Global	0	2	String	CollectPositionData	z
			numString	IntCallP	Global	1	1	String	CollectPositionData	numString
			declLocation	DataBase	Dereferen	0	1	positionStackPtr	CollectPositionData	-
			numString	IntCallP	Global	0	1	StatsWrite	WriteSCTitles	str
			z	IntCallP	Global	0	2	String	CollectPositionData	z
			numString	IntCallP	Global	1	1	String	CollectPositionData	numString
			dProcLevel	DataBase	Dereferen	0	1	positionStackPtr	CollectPositionData	-
			numString	IntCallP	Global	0	1	StatsWrite	WriteSCTitles	str
			z	IntCallP	Global	0	1	String	CollectPositionData	z
			numString	IntCallP	Global	0	1	String	CollectPositionData	numString
			statsFile	IntCallP	External	0	1	Stream	CollectPositionData	statsFile
			CR	IntCallP	Imported	0	1	Stream	CollectPositionData	Ascii
			aSP	FmlParam	-	0	0	-	-	-
			fSP	FmlParam	-	0	0	-	-	-
			SLC	FmlParam	-	0	0	-	-	-
			found	FmlParam	-	0	0	-	-	-
			fullCount	DataBase	Global	2	9	fullCount	StructChartImpl	-
			maxStack	DataBase	Imported	0	1	Display	-	-
			count	DataBase	Local	1	0	-	-	-
			maxStack	IntCallP	Imported	0	1	Utils	UpdateFullStack	Display
			count	IntCallP	Local	0	1	log	Exit	count
			found	DataBase	UsefmlCal	0	1	-	-	-
			fullStackPtr	DataBase	Global	0	6	fullStackPtr	StructChartImpl	-
declLocation	DataBase	Dereferen	1	0	fullStackPtr	UpdateFullStack	-			
positionStackPtr	DataBase	Global	0	1	positionStackPtr	StructChartImpl	-			
stackIndex	DataBase	Global	0	2	stackIndex	StructChartImpl	-			
declLocation	DataBase	Dereferen	0	1	positionStackPtr	UpdateFullStack	-			
fileNumber	DataBase	Local	1	1	-	-	-			
fSP	IntCallP	UsefmlCal	0	1	FindFileNumber	CheckReturnConditio	fSP			
posIndex	IntCallP	Global	0	1	FindFileNumber	CheckReturnConditio	callLocation			
callLocation	DataBase	Dereferen	1	0	fullStackPtr	UpdateFullStack	-			
posIndex	DataBase	Global	0	2	posIndex	StructChartImpl	-			
callLevel	DataBase	Dereferen	1	0	fullStackPtr	UpdateFullStack	-			
fileNumber	DataBase	Dereferen	1	0	fullStackPtr	UpdateFullStack	-			
SLC	DataBase	UsefmlCal	0	1	-	-	-			
SLC	DataBase	Dereferen	1	0	fullStackPtr	UpdateFullStack	-			
declLocation	DataBase	Dereferen	1	0	fullStackPtr	UpdateFullStack	-			
6	SuppressOutput	nonT	aSP	FmlParam	-	0	0	-	-	-
			debugg	FmlParam	-	0	0	-	-	-
			posIndex	DataBase	Global	0	1	posIndex	StructChartImpl	-
			startingPLevel	DataBase	Local	1	1	-	-	-
			aSP	DataBase	UsefmlCal	0	2	-	-	-
			tempIndex	DataBase	Local	1	4	-	-	-
			Plevel	DataBase	Dereferen	0	1	aSP	SuppressOutput	-
			debugg	DataBase	UsefmlCal	0	2	-	-	-
			z	IntCallP	Global	0	2	String	CollectPositionData	z
			numString	IntCallP	Global	1	1	String	CollectPositionData	numString
			procLevel	DataBase	Dereferen	0	1	positionStackPtr	CollectPositionData	-
			numString	IntCallP	Global	0	1	StatsWrite	WriteSCTitles	str
			z	IntCallP	Global	0	2	String	CollectPositionData	z
			numString	IntCallP	Global	1	1	String	CollectPositionData	numString

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION				
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias	
6	SearchForDeclaratio	nonT	numString	DataBase	Global	1	0	numString	StructChartImpl	-	
			posIndex	IntCallip	Global	1	1	Utils	SuppressOutput	posIndex	
			z	DataBase	Global	0	1	z	StructChartImpl	-	
			maxIndexLength	IntCallip	Global	0	1	CheckLength	StatsWrite	length	
			numString	IntCallip	Global	0	1	StatsWrite	WritesCTitles	str	
			z	IntCallip	Global	0	2	String	SuppressOutput	z	
			numString	IntCallip	Global	1	1	String	SuppressOutput	numString	
			Plevel	DataBase	Dereferen	0	1	aSP	SuppressOutput	-	
			posIndex	IntCallip	Global	0	1	Utils	SuppressOutput	posIndex	
			maxIndexLength	IntCallip	Global	0	1	CheckLength	StatsWrite	length	
			numString	IntCallip	Global	0	1	StatsWrite	WritesCTitles	str	
			z	IntCallip	Global	0	1	String	SuppressOutput	z	
			numString	IntCallip	Global	0	1	String	SuppressOutput	numString	
			statsFile	IntCallip	External	0	1	Stream	SuppressOutput	statsFile	
			CR	IntCallip	Imported	0	1	Stream	SuppressOutput	Ascii	
			aSP	FmlParam	-	-	0	0	-	-	-
			fSP	FmlParam	-	-	0	0	-	-	-
			MaxFiles	FmlParam	-	-	0	0	-	-	-
			tempFileCount	DataBase	Local	3	6	-	-	-	-
			MaxFiles	UseFmlCal	UseFmlCal	0	1	-	-	-	-
			fSP	DataBase	UseFmlCal	0	4	-	-	-	-
			fileCount	DataBase	Dereferen	0	1	fSP	SearchForDeclaratio	-	-
			Equal	DataBase	Imported	0	2	String	-	-	-
			aSP	DataBase	UseFmlCal	0	7	-	-	-	-
			posIndex	DataBase	Global	1	5	posIndex	StructChartImpl	-	-
			IEatom	DataBase	Dereferen	0	1	aSP	SearchForDeclaratio	-	-
			exportName	DataBase	Dereferen	0	1	fSP	SearchForDeclaratio	-	-
			numString	DataBase	Global	1	0	numString	StructChartImpl	-	-
			CopyToNewString	DataBase	Imported	0	1	String	-	-	-
			inputString	DataBase	Dereferen	0	1	aSP	SearchForDeclaratio	-	-
			z	DataBase	Global	0	1	z	StructChartImpl	-	-
			found	DataBase	Local	1	3	-	-	-	-
			aSP	IntCallip	UseFmlCal	0	1	CheckProcExists	ReduceLength	aSP	aSP
			fSP	IntCallip	UseFmlCal	0	1	CheckProcExists	ReduceLength	fSP	fSP
			tempFileCount	IntCallip	Local	0	1	CheckProcExists	ReduceLength	tempFileCount	tempFileCount
			numString	IntCallip	Global	0	1	CheckProcExists	ReduceLength	numString	numString
			z	IntCallip	Global	0	1	String	SearchForDeclaratio	z	z
			numString	IntCallip	Global	0	1	String	SearchForDeclaratio	numString	numString
			callLocation	DataBase	Local	1	0	-	-	-	-
			Index	DataBase	Dereferen	0	1	aSP	SearchForDeclaratio	-	-
			fSP	IntCallip	UseFmlCal	0	1	FindFileNumber	CheckReturnConditio	fSP	fSP
			callLocation	IntCallip	Local	0	1	FindFileNumber	CheckReturnConditio	callLocation	callLocation
			tempIndex	DataBase	Local	2	6	-	-	-	-
			startIndex	DataBase	Dereferen	0	1	fSP	SearchForDeclaratio	-	-
			endIndex	DataBase	Dereferen	0	1	fSP	SearchForDeclaratio	-	-
			idMark	DataBase	Dereferen	0	1	aSP	SearchForDeclaratio	-	-
			procedure	DataBase	Local	0	1	-	-	-	-
			inputString	DataBase	Dereferen	0	1	aSP	SearchForDeclaratio	-	-
			inputString	DataBase	Dereferen	0	1	aSP	SearchForDeclaratio	-	-
			declLocation	DataBase	Dereferen	1	0	aSP	SearchForDeclaratio	-	-

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE				DATA				CONNECTION			
Lvl	Node Name	Type	Term	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
6	GoFindCallParams	Term		found	RetVal	Local	0	1	FindFileNumber	CheckReturnConditio	found
				aIP	FmlParam	-	0	0	-	-	-
				aSP	FmlParam	-	0	0	-	-	-
				callIndex	FmlParam	-	0	0	-	-	-
				fullStackPtr	DataBase	Local	3	10	-	-	-
				fullStackPtr	DataBase	UseFmlCal	0	2	-	-	-
				fullCount	DataBase	Global	0	2	fullCount	StructChartImpl	-
				callLocation	DataBase	Dereferen	0	1	fullStackPtr	GoFindCallParams	-
				aSP	DataBase	UseFmlCal	0	13	-	-	-
				listMark	DataBase	Dereferen	0	1	aSP	GoFindCallParams	-
				callParameter	DataBase	Global	0	2	callParameter	StructChartImpl	-
				symbolNumber	DataBase	Dereferen	0	1	aSP	GoFindCallParams	-
				semiColon	DataBase	Global	0	4	semiColon	StructChartImpl	-
				symbolNumber	DataBase	Dereferen	0	1	aSP	GoFindCallParams	-
				equals	DataBase	Global	0	2	equals	StructChartImpl	-
				listMark	DataBase	Dereferen	0	1	aSP	GoFindCallParams	-
				symbolNumber	DataBase	Dereferen	0	1	aSP	GoFindCallParams	-
				symbolNumber	DataBase	Dereferen	0	1	aSP	GoFindCallParams	-
				idMark	DataBase	Dereferen	0	1	aSP	GoFindCallParams	-
				variable	DataBase	Global	0	2	variable	StructChartImpl	-
				paramCount	DataBase	Local	1	2	-	-	-
				declIndex	DataBase	Local	3	8	-	-	-
				declLocation	DataBase	Dereferen	0	1	fullStackPtr	GoFindCallParams	-
				declParamCount	DataBase	Local	2	2	-	-	-
				listMark	DataBase	Dereferen	0	1	aSP	GoFindCallParams	-
				parameter	DataBase	Global	0	2	parameter	StructChartImpl	-
				symbolNumber	DataBase	Dereferen	0	1	aSP	GoFindCallParams	-
				listMark	DataBase	Dereferen	0	1	aSP	GoFindCallParams	-
				symbolNumber	DataBase	Dereferen	0	1	aSP	GoFindCallParams	-
				idMark	DataBase	Dereferen	0	1	aSP	GoFindCallParams	-
				declLocation	DataBase	Dereferen	1	0	aSP	GoFindCallParams	-
6	StudyRepeats	Term		aSP	FmlParam	-	0	0	-	-	-
				mainLoopCount	DataBase	Local	1	9	-	-	-
				fullCount	DataBase	Global	0	1	fullCount	StructChartImpl	-
				currentCall	DataBase	Local	1	2	-	-	-
				fullStackPtr	DataBase	Global	0	13	fullStackPtr	StructChartImpl	-
				callLocation	DataBase	Dereferen	0	1	fullStackPtr	StudyRepeats	-
				currentLevel	DataBase	Local	1	3	-	-	-
				callLevel	DataBase	Dereferen	0	1	fullStackPtr	StudyRepeats	-
				localLoopCount	DataBase	Local	4	12	-	-	-
				callLevel	DataBase	Dereferen	0	1	fullStackPtr	StudyRepeats	-
				callLevel	DataBase	Dereferen	0	1	fullStackPtr	StudyRepeats	-
				comparisonCall	DataBase	Local	2	2	-	-	-
				callLocation	DataBase	Dereferen	0	1	fullStackPtr	StudyRepeats	-
				Equal	DataBase	Imported	0	2	String	-	-
				aSP	DataBase	UseFmlCal	0	4	-	-	-
				inputString	DataBase	Dereferen	0	1	aSP	StudyRepeats	-
				inputString	DataBase	Dereferen	0	1	aSP	StudyRepeats	-
				twInRepeat	DataBase	Dereferen	1	0	fullStackPtr	StudyRepeats	-

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE				DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias	
6	WritesCTitles	nonT	firstOfTwin	DataBase	Dereferen	1	0	fullStackPtr	StudyRepeats	-	
			callLocation	DataBase	Dereferen	0	1	fullStackPtr	StudyRepeats	-	
			comparisonLevel	DataBase	Local	1	1	-	-	-	
			callLevel	DataBase	Dereferen	0	1	fullStackPtr	StudyRepeats	-	
			inputString	DataBase	Dereferen	0	1	aSP	StudyRepeats	-	
			inputString	DataBase	Dereferen	0	1	aSP	StudyRepeats	-	
			globalRepeat	DataBase	Dereferen	1	0	fullStackPtr	StudyRepeats	-	
			firstOfGlobal	DataBase	Dereferen	1	0	fullStackPtr	StudyRepeats	-	
			levelRepeat	DataBase	Dereferen	1	0	fullStackPtr	StudyRepeats	-	
			firstOfLevel	DataBase	Dereferen	1	0	fullStackPtr	StudyRepeats	-	
6	SemiGraphOutput	nonT	statsFile	IntCallP	External	0	1	Stream	WritesCTitles	statsFile	
			CR	IntCallP	Imported	0	1	Stream	WritesCTitles	Ascii	
			statsFile	IntCallP	External	0	1	Stream	WritesCTitles	statsFile	
			CR	IntCallP	Imported	0	1	Stream	WritesCTitles	Ascii	
			statsFile	IntCallP	External	0	1	Stream	WritesCTitles	statsFile	
			CR	IntCallP	Imported	0	1	Stream	WritesCTitles	Ascii	
			statsFile	IntCallP	External	0	1	Stream	WritesCTitles	statsFile	
			CR	IntCallP	Imported	0	1	Stream	WritesCTitles	Ascii	
			lineCount	DataBase	Global	1	1	lineCount	WritesCTitles	Ascii	
			titleSize	DataBase	Local	0	1	-	StructChartImpl	-	
6	SemiGraphOutput	nonT	aSP	FmlParam	-	0	0	-	-	-	
			fSP	FmlParam	-	0	0	-	-	-	
			debugg	FmlParam	-	0	0	-	-	-	
			repeats	FmlParam	-	0	0	-	-	-	
			tempCount	DataBase	Local	1	6	-	-	-	
			fullCount	DataBase	Global	0	2	fullCount	StructChartImpl	-	
			tempLevel	DataBase	Local	2	2	-	-	-	
			currentCall	DataBase	Local	1	4	-	-	-	
			fullStackPtr	DataBase	Global	0	6	fullStackPtr	StructChartImpl	-	
			callLocation	DataBase	Dereferen	0	1	fullStackPtr	SemiGraphOutput	-	
6	SemiGraphOutput	nonT	currentLevel	DataBase	Local	1	0	-	-	-	
			callLevel	DataBase	Dereferen	0	1	fullStackPtr	SemiGraphOutput	-	
			SLC	DataBase	Local	1	1	-	-	-	
			SLC	DataBase	Dereferen	0	1	fullStackPtr	SemiGraphOutput	-	
			twinRepeat	DataBase	Dereferen	0	1	fullStackPtr	SemiGraphOutput	-	
			repeats	DataBase	Dereferen	0	1	fullStackPtr	SemiGraphOutput	-	
			numString	DataBase	UseFmlCal	0	1	-	-	-	
			tempCount	DataBase	Global	1	0	numString	StructChartImpl	-	
			z	IntCallP	Local	1	5	Utils	SemiGraphOutput	tempCount	
			maxIndexLength	IntCallP	Global	0	1	z	StructChartImpl	-	
6	SemiGraphOutput	nonT	numString	IntCallP	Global	0	1	CheckLength	StatsWrite	length	
			z	IntCallP	Global	0	1	StatsWrite	WritesCTitles	str	
			numString	IntCallP	Global	0	2	String	SemiGraphOutput	z	
			fileNumber	IntCallP	Global	1	1	String	SemiGraphOutput	numString	
			fileNumber	DataBase	Local	2	2	-	-	-	
			CopyToNewString	DataBase	Dereferen	0	1	fullStackPtr	SemiGraphOutput	-	
			fSP	DataBase	Imported	0	4	String	-	-	
				DataBase	UseFmlCal	0	2	-	-	-	

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE				DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias	
6	TabularOutput	nonT	fileName	DataBase	Dereferen	0	1	fSP	SemiGraphOutput	-	
			maxFileLength	IntCallp	Global	0	1	CheckLength	StatsWrite	length	
			numString	IntCallp	Global	0	1	StatsWrite	WriteSCITitles	str	
			z	IntCallp	Global	0	2	String	SemiGraphOutput	z	
			numString	IntCallp	Global	1	1	String	SemiGraphOutput	numString	
			aSP	DataBase	UseFmlCal	0	4	-	-	-	
			IEatom	DataBase	Dereferen	0	1	aSP	SemiGraphOutput	-	
			maxIEidentLength	IntCallp	Global	0	1	CheckLength	StatsWrite	length	
			numString	IntCallp	Global	0	1	StatsWrite	WriteSCITitles	str	
			z	IntCallp	Global	0	1	String	SemiGraphOutput	z	
			numString	IntCallp	Global	2	1	String	SemiGraphOutput	numString	
			currentDecl	IntCallp	Local	1	1	String	-	-	
			declLocation	DataBase	Dereferen	0	1	aSP	SemiGraphOutput	-	
			deref	DataBase	Local	1	1	-	-	-	
			varScope	DataBase	Dereferen	0	1	aSP	SemiGraphOutput	-	
			z	IntCallp	Global	0	2	String	SemiGraphOutput	z	
			fSP	IntCallp	UseFmlCal	0	1	FindFileNumber	CheckReturnConditio	fSP	
			currentDecl	IntCallp	Local	0	1	FindFileNumber	CheckReturnConditio	callLocation	
			fileName	DataBase	Dereferen	0	1	fSP	SemiGraphOutput	-	
			maxFileLength	IntCallp	Global	0	1	CheckLength	StatsWrite	length	
			numString	IntCallp	Global	0	1	StatsWrite	WriteSCITitles	str	
			z	IntCallp	Global	0	2	String	SemiGraphOutput	z	
			numString	IntCallp	Global	1	1	String	SemiGraphOutput	numString	
			currentLevel	IntCallp	Local	0	3	Utils	SemiGraphOutput	currentLevel	
			numString	IntCallp	Global	0	1	StatsWrite	WriteSCITitles	str	
			z	IntCallp	Global	0	2	String	SemiGraphOutput	z	
			numString	IntCallp	Global	1	1	String	SemiGraphOutput	numString	
			inputString	IntCallp	Global	0	1	aSP	SemiGraphOutput	-	
			numString	IntCallp	Global	0	1	StatsWrite	WriteSCITitles	str	
			z	IntCallp	Global	0	1	String	SemiGraphOutput	z	
			numString	IntCallp	Global	0	1	String	SemiGraphOutput	numString	
			statsFile	IntCallp	Global	0	1	Stream	SemiGraphOutput	statsFile	
			CR	IntCallp	External	0	1	Stream	SemiGraphOutput	Ascii	
			lineCount	IntCallp	Imported	0	1	Stream	SemiGraphOutput	-	
			pageSize	IntCallp	Global	2	4	lineCount	SemiGraphOutput	-	
			callLevel	DataBase	Global	0	2	pageSize	StructChartImpl	-	
			statsFile	DataBase	Dereferen	0	1	fullStackPtr	StructChartImpl	-	
			CR	IntCallp	External	0	1	Stream	SemiGraphOutput	statsFile	
			nonT	IntCallp	Imported	0	1	Stream	SemiGraphOutput	Ascii	
			aSP	FmlParam	-	0	0	-	-	-	
			fSP	FmlParam	-	0	0	-	-	-	
			debugg	FmlParam	-	0	0	-	-	-	
			repeats	FmlParam	-	0	0	-	-	-	
			lineCount	DataBase	Global	2	3	lineCount	StructChartImpl	-	
			titleSize	DataBase	Local	0	1	-	-	-	
			tempStackCount	DataBase	Local	3	9	-	-	-	
			fullCount	DataBase	Global	0	2	fullCount	StructChartImpl	-	
			fullStackPtr	DataBase	Global	0	5	fullStackPtr	StructChartImpl	-	
			SLC	DataBase	Dereferen	0	1	fullStackPtr	TabularOutput	-	
			SLC	DataBase	Local	3	1	-	-	-	
			tempStackCount	IntCallp	Local	0	1	OutputTableLine	TabularOutput	tempStackCount	

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE				DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias	
6	Text GetTableBase ExpandStack	Term Term nonT	fSP	IntCallP	UseFmlCal	0	1	OutputTableLine	TabularOutput	fSP	
			aSP	IntCallP	UseFmlCal	0	1	OutputTableLine	TabularOutput	aSP	
			SLC	IntCallP	Local	0	1	OutputTableLine	TabularOutput	SLC	
			titleSize	IntCallP	Local	0	1	OutputTableLine	TabularOutput	titleSize	
			currentLevel	DataBase	Local	1	3	-	-	-	
			maxLevels	DataBase	Global	0	1	maxLevels	StructChartImpl	-	
			SLC	DataBase	Dereferen	0	1	fullStackPtr	TabularOutput	-	
			callLevel	DataBase	Dereferen	0	1	fullStackPtr	TabularOutput	-	
			repeats	DataBase	UseFmlCal	0	1	-	-	-	
			tempStackCount	IntCallP	Local	0	1	OutputTableLine	TabularOutput	tempStackCount	
			fSP	IntCallP	UseFmlCal	0	1	OutputTableLine	TabularOutput	fSP	
			aSP	IntCallP	UseFmlCal	0	1	OutputTableLine	TabularOutput	aSP	
			SLC	IntCallP	Local	0	1	OutputTableLine	TabularOutput	SLC	
			titleSize	IntCallP	Local	0	1	OutputTableLine	TabularOutput	titleSize	
			twinRepeat	IntCallP	Local	0	1	OutputTableLine	TabularOutput	-	
6	ExpandQueue	nonT	levelRepeat	DataBase	Dereferen	0	1	fullStackPtr	TabularOutput	-	
			tempStackCount	IntCallP	Local	0	1	OutputTableLine	TabularOutput	tempStackCount	
			fSP	IntCallP	UseFmlCal	0	1	OutputTableLine	TabularOutput	fSP	
			aSP	IntCallP	UseFmlCal	0	1	OutputTableLine	TabularOutput	aSP	
			SLC	IntCallP	Local	0	1	OutputTableLine	TabularOutput	SLC	
			titleSize	IntCallP	Local	0	1	OutputTableLine	TabularOutput	titleSize	
			pageSize	DataBase	Global	0	1	pageSize	StructChartImpl	-	
			statsFile	IntCallP	External	0	1	Stream	TabularOutput	statsFile	
			CR	IntCallP	Imported	0	1	Stream	TabularOutput	Ascii	
			delta	FmlParam	-	0	0	-	-	-	
			s	DataBase	Global	1	2	s	ParserImpl	-	
			length	DataBase	Imported	0	1	s	-	-	
			oldSize	DataBase	Local	0	2	-	-	-	
			delta	DataBase	UseFmlCal	0	1	-	-	-	
			i	DataBase	Local	0	6	-	-	-	
6	EraseQueue	Term	l	DataBase	Global	1	1	l	ParserImpl	-	
			v	DataBase	Global	1	1	v	ParserImpl	-	
			delta	FmlParam	-	0	0	-	-	-	
			q	DataBase	Global	1	2	q	ParserImpl	-	
			length	DataBase	Imported	0	1	q	-	-	
			oldSize	DataBase	Local	0	2	-	-	-	
			delta	DataBase	UseFmlCal	0	1	-	-	-	
			i	DataBase	Local	0	2	-	-	-	
			q	DataBase	Global	0	2	q	ParserImpl	-	
			s	DataBase	Global	0	2	s	ParserImpl	-	
			v	DataBase	Global	0	1	v	ParserImpl	-	
			l	DataBase	Global	0	1	l	ParserImpl	-	
			str	FmlParam	-	0	0	-	-	-	
			statsWrite	-	-	-	-	-	-	-	
			AssignDescriptors	-	-	-	-	-	-	-	

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA					CONNECTION		
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
7	UserAbort	Term	tempIndex	DataBase	Local	1	3	-	-	-
	CopyToNewString	Term	eSP	DataBase	UseFmlCal	0	1	-	-	-
	CardToString	Term	missingProcLocation	DataBase	Dereferen	0	1	eSP	OutputExportInfo	-
	log	nonT	CopyToNewString	DataBase	Imported	0	3	String	-	-
			aSP	DataBase	UseFmlCal	0	3	-	-	-
			IEatom	DataBase	Dereferen	0	1	aSP	OutputExportInfo	-
			numString	IntCallP	Global	0	1	String	OutputExportInfo	numString
			maxExportNameLength	IntCallP	Local	0	1	CheckLength	StatsWrite	length
			numString	IntCallP	Global	0	1	StatsWrite	WritesCTitles	str
			z	IntCallP	Global	0	2	String	OutputExportInfo	z
7	FindFileNumber	Term	numString	IntCallP	Global	1	1	String	OutputExportInfo	numString
			inputString	DataBase	Dereferen	0	1	aSP	OutputExportInfo	-
			maxFileNameLength	IntCallP	Local	0	1	CheckLength	StatsWrite	length
			numString	IntCallP	Global	0	1	StatsWrite	WritesCTitles	str
			z	IntCallP	Global	0	1	String	OutputExportInfo	z
			numString	IntCallP	Global	0	1	String	OutputExportInfo	numString
			statsFile	IntCallP	External	0	1	Stream	OutputExportInfo	statsFile
			CR	IntCallP	Imported	0	1	Stream	OutputExportInfo	statsFile
			CR	IntCallP	Imported	0	1	Stream	OutputExportInfo	statsFile
			fileSW	IntCallP	Public	0	1	Put	log	fileSW
7	FindFileNumber	Term	s	IntCallP	Imported	0	1	Put	log	-
			fSP	FmlParam	-	0	0	-	-	-
			callLocation	FmlParam	-	0	0	-	-	-
			notFound	DataBase	Local	1	1	-	-	-
			callLocation	DataBase	UseFmlCal	0	2	-	-	-
			fSP	DataBase	UseFmlCal	0	2	-	-	-
			fileCount	DataBase	Local	1	3	-	-	-
			startIndex	DataBase	Dereferen	0	1	fSP	FindFileNumber	-
			endIndex	DataBase	Dereferen	0	1	fSP	FindFileNumber	-
			fileCount	RetVal	Local	0	1	Stream	CheckReturnConditio	fileCount
7	WritesCTitles	nonT	statsFile	IntCallP	External	0	1	Stream	WritesCTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WritesCTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WritesCTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WritesCTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WritesCTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WritesCTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WritesCTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WritesCTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WritesCTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WritesCTitles	Ascii

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
7	WriteTableTitles	nonT	statsFile	IntCallP	External	0	1	Stream	WriteSCITitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteSCITitles	Ascii
			lineCount	DataBase	Global	1	1	lineCount	StructChartImpl	-
			titleSize	DataBase	Local	0	1	-	-	-
			statsFile	IntCallP	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteTableTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteTableTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteTableTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteTableTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteTableTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteTableTitles	Ascii
7	OutputTableLine	nonT	tempStackCount	FmlParam	-	0	0	-	-	-
			fSP	FmlParam	-	0	0	-	-	-
			ASP	FmlParam	-	0	0	-	-	-
			SLC	FmlParam	-	0	0	-	-	-
			titleSize	FmlParam	-	0	0	-	-	-
			tempStackCount	DataBase	UsefmlCal	0	1	-	-	-
			fullStackPtr	DataBase	Global	0	7	fullStackPtr	StructChartImpl	-
			tSC	DataBase	Local	0	1	-	-	-
			callLevel	DataBase	Dereferen	0	1	fullStackPtr	OutputTableLine	-
			SLC	DataBase	Dereferen	0	1	fullStackPtr	WriteRepeatInfo	-
			numString	DataBase	Dereferen	0	1	fullStackPtr	StructChartImpl	-
			level	IntCallP	Local	0	3	Utils	WriteRepeatInfo	level
			maxLevelLength	IntCallP	Global	0	1	CheckLength	StatsWrite	length
			numString	IntCallP	Global	0	1	StatsWrite	WriteSCITitles	str
			z	IntCallP	Global	0	2	String	WriteRepeatInfo	z
			numString	IntCallP	Global	1	1	String	WriteRepeatInfo	numString
			callLocation	DataBase	Local	2	4	-	-	-
			callLocation	DataBase	Dereferen	0	1	fullStackPtr	WriteRepeatInfo	-
			CopyToNewString	DataBase	Imported	0	6	String	-	-
			ASP	DataBase	UsefmlCal	0	4	-	-	-
			inputString	DataBase	Dereferen	0	1	ASP	WriteRepeatInfo	-
			maxProcLength	IntCallP	Global	0	1	CheckLength	StatsWrite	length
			numString	IntCallP	Global	0	1	StatsWrite	WriteSCITitles	str
			z	IntCallP	Global	0	2	String	WriteRepeatInfo	z
			numString	IntCallP	Global	1	1	String	WriteRepeatInfo	numString
			IEatom	DataBase	Dereferen	0	1	ASP	WriteRepeatInfo	-
			maxIEidentLength	IntCallP	Global	0	1	CheckLength	StatsWrite	length
			numString	IntCallP	Global	0	1	StatsWrite	WriteSCITitles	str
			z	IntCallP	Global	0	2	String	WriteRepeatInfo	z

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
	numString		numString	IntCallip	Global	1	1	String	WriteRepeatInfo	numString
	fileNumber		fileNumber	DataBase	Local	2	2	-	-	-
	fSP		fSP	IntCallip	UseFmlCal	0	1	FindFileNumber	CheckReturnConditio	fSP
	callLocation		callLocation	IntCallip	Local	0	1	FindFileNumber	CheckReturnConditio	callLocation
	fSP		fSP	DataBase	UseFmlCal	0	2	-	-	-
	fileName		fileName	DataBase	Dereferen	0	1	fSP	WriteRepeatInfo	-
	size		size	DataBase	Local	2	0	-	-	-
	numString		numString	IntCallip	Global	0	1	String	WriteRepeatInfo	numString
	size		size	IntCallip	Local	0	1	ReduceLength	AddSpaces	size
	maxFileLength		maxFileLength	IntCallip	Local	0	1	ReduceLength	AddSpaces	length
	numString		numString	IntCallip	Global	0	1	CheckLength	StatsWrite	length
	z		z	IntCallip	Global	0	1	StatsWrite	WriteSCITitles	str
	numString		numString	IntCallip	Global	0	1	String	WriteRepeatInfo	z
	callLevel		callLevel	IntCallip	Global	1	1	String	WriteRepeatInfo	numString
	callLevel		callLevel	DataBase	Dereferen	0	1	fullStackPtr	WriteRepeatInfo	-
	twinRepeat		twinRepeat	DataBase	Dereferen	0	1	fullStackPtr	WriteRepeatInfo	-
	firstLine		firstLine	DataBase	Dereferen	0	1	fullStackPtr	WriteRepeatInfo	-
	lineCount		lineCount	DataBase	Local	1	1	-	-	-
	statsFile		statsFile	DataBase	Global	4	6	lineCount	StructChartImpl	-
	CR		CR	IntCallip	External	0	1	Stream	WriteRepeatInfo	statsFile
	pageSize		pageSize	IntCallip	Imported	0	1	Stream	WriteRepeatInfo	Ascii
	titleSize		titleSize	DataBase	Global	0	2	pageSize	StructChartImpl	-
	z		z	DataBase	UseFmlCal	0	2	-	-	-
	maxSubMargin		maxSubMargin	IntCallip	Global	0	1	String	WriteRepeatInfo	z
	numString		numString	IntCallip	Global	0	1	CheckLength	StatsWrite	length
	z		z	IntCallip	Global	0	1	StatsWrite	WriteSCITitles	str
	numString		numString	IntCallip	Global	0	2	String	WriteRepeatInfo	z
	level		level	IntCallip	Global	1	1	String	WriteRepeatInfo	numString
	numString		numString	IntCallip	Local	0	1	Utils	WriteRepeatInfo	level
	maxLevellLength		maxLevellLength	IntCallip	Global	0	1	StatsWrite	WriteSCITitles	str
	z		z	IntCallip	Global	0	1	CheckLength	StatsWrite	length
	numString		numString	IntCallip	Global	0	2	String	WriteRepeatInfo	z
	callLocation		callLocation	IntCallip	Global	1	1	String	WriteRepeatInfo	numString
	inputString		inputString	DataBase	Dereferen	0	1	fullStackPtr	WriteRepeatInfo	-
	maxProclLength		maxProclLength	DataBase	Dereferen	0	1	asp	WriteRepeatInfo	-
	numString		numString	IntCallip	Global	0	1	CheckLength	StatsWrite	length
	z		z	IntCallip	Global	0	1	StatsWrite	WriteSCITitles	str
	numString		numString	IntCallip	Global	0	2	String	WriteRepeatInfo	z
	IEatom		IEatom	IntCallip	Global	1	1	String	WriteRepeatInfo	numString
	maxIIdentLength		maxIIdentLength	DataBase	Dereferen	0	1	asp	WriteRepeatInfo	-
	numString		numString	IntCallip	Global	0	1	CheckLength	StatsWrite	length
	z		z	IntCallip	Global	0	1	StatsWrite	WriteSCITitles	str
	numString		numString	IntCallip	Global	0	2	String	WriteRepeatInfo	z
	fSP		fSP	IntCallip	Global	1	1	String	WriteRepeatInfo	numString
	callLocation		callLocation	IntCallip	UseFmlCal	0	1	FindFileNumber	CheckReturnConditio	fSP
	fileName		fileName	IntCallip	Local	0	1	FindFileNumber	CheckReturnConditio	callLocation
	numString		numString	DataBase	Dereferen	0	1	fSP	WriteRepeatInfo	-
	size		size	IntCallip	Dereferen	0	1	String	WriteRepeatInfo	numString
	size		size	IntCallip	Global	0	1	ReduceLength	AddSpaces	size
	maxFileLength		maxFileLength	IntCallip	Local	0	1	ReduceLength	AddSpaces	length
				IntCallip	Local	0	1	CheckLength	StatsWrite	length

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
7	EraseStack	Term	numString	IntCallP	Global	0	1	StatsWrite	WriteSCTitles	str
			z	IntCallP	Global	0	1	String	WriteRepeatInfo	z
			numString	IntCallP	Global	0	1	String	WriteRepeatInfo	numString
			statsFile	IntCallP	External	0	1	Stream	WriteRepeatInfo	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteRepeatInfo	Ascii
7	AssignDescriptors	Term	s	DataBase	Global	0	2	s	ParserImpl	-
			v	DataBase	Global	0	1	v	ParserImpl	-
			1	DataBase	Global	0	1	1	ParserImpl	-
8	PutString	Term	q	DataBase	Global	0	2	q	ParserImpl	-
8	AddSpaces	nonT	size	FmlParam	-	0	0	-	-	-
			length	FmlParam	-	0	0	-	-	-
			size	DataBase	UseFmlCal	2	3	-	-	-
			length	DataBase	UseFmlCal	0	1	-	-	-
			numString	IntCallP	Global	0	1	String	AddSpaces	numString
B	ReduceLength	nonT	z	IntCallP	Global	0	1	String	AddSpaces	z
			size	FmlParam	-	0	0	-	-	-
			length	FmlParam	-	0	0	-	-	-
			subStringDescr	DataBase	Local	1	0	-	-	-
			numString	DataBase	Global	0	1	numString	StructChartImpl	-
			length	DataBase	UseFmlCal	0	1	-	-	-
			tempString	DataBase	Local	1	0	-	-	-
			z	IntCallP	Global	0	1	String	ReduceLength	z
			length	IntCallP	UseFmlCal	0	1	String	ReduceLength	length
			tempString	IntCallP	Local	0	1	String	ReduceLength	tempString
			subStringDescr	IntCallP	Local	0	1	String	ReduceLength	subStringDescr
			z	IntCallP	Global	0	1	String	ReduceLength	z
			numString	IntCallP	Global	1	1	String	ReduceLength	numString
			tempString	IntCallP	Local	0	1	String	ReduceLength	tempString
8	WriteExportTitles	nonT	z	IntCallP	Global	0	1	String	ReduceLength	z
			tempString	IntCallP	Global	0	1	String	ReduceLength	tempString
			z	IntCallP	Local	0	1	String	ReduceLength	z
			tempString	IntCallP	Local	0	1	String	ReduceLength	tempString
			statsFile	IntCallP	External	0	1	Stream	WriteExportTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteExportTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WriteExportTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteExportTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WriteExportTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteExportTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WriteExportTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteExportTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WriteExportTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteExportTitles	Ascii
			statsFile	IntCallP	External	0	1	Stream	WriteExportTitles	statsFile

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
6 8 8	CopyToNewString CardToString CheckLength	Term Term nonT	statsFile	IntCallP	External	0	1	Stream	WriteExportTitles	statsFile
			CR	IntCallP	Imported	0	1	Stream	WriteExportTitles	Ascii
			length	FmlParam	-	0	0	-	-	-
			size	DataBase	Local	1	2	-	-	-
			numString	IntCallP	Global	0	1	String	CheckLength	numString
8 8 8 8	StatsWrite	nonT	length	DataBase	UsefmlCal	0	2	-	-	-
			size	IntCallP	Local	0	1	AddSpaces	CheckLength	size
			length	IntCallP	UsefmlCal	0	1	AddSpaces	CheckLength	length
			size	IntCallP	Local	0	1	ReduceLength	AddSpaces	size
			length	IntCallP	UsefmlCal	0	1	ReduceLength	AddSpaces	length
8 8 8 8	FreeString Equal PutChar Text WriteRepeatInfo	Term Term Term nonT	str	FmlParam	-	0	0	-	-	-
			statsFile	IntCallP	External	0	1	Stream	StatsWrite	statsFile
			str	IntCallP	UsefmlCal	0	1	Stream	StatsWrite	str
			numString	DataBase	UsefmlCal	1	0	-	-	-
			z	IntCallP	Global	0	1	String	WriteRepeatInfo	z
8	FindFileNumber	Term	fullStackPtr	DataBase	UsefmlCal	0	6	-	-	-
			tSC	DataBase	Local	2	14	-	-	-
			firstDfTwin	DataBase	Dereferen	0	1	fullStackPtr	WriteRepeatInfo	-
			twinRepeat	DataBase	Dereferen	0	1	fullStackPtr	WriteRepeatInfo	-
			numString	IntCallP	UsefmlCal	0	1	String	WriteRepeatInfo	numString
			z	IntCallP	Global	0	1	String	WriteRepeatInfo	z
			firstDfLevel	DataBase	Dereferen	0	1	fullStackPtr	WriteRepeatInfo	-
			levelRepeat	DataBase	Dereferen	0	1	fullStackPtr	WriteRepeatInfo	-
			numString	IntCallP	UsefmlCal	0	1	String	WriteRepeatInfo	numString
			z	IntCallP	Global	0	1	String	WriteRepeatInfo	z
			firstDfGlobal	DataBase	Dereferen	0	1	fullStackPtr	WriteRepeatInfo	-
			globalRepeat	DataBase	Dereferen	0	1	fullStackPtr	WriteRepeatInfo	-
			numString	IntCallP	UsefmlCal	0	1	String	WriteRepeatInfo	numString
			z	IntCallP	Global	0	1	String	WriteRepeatInfo	z
			maxRepeatLength	IntCallP	Global	0	1	CheckLength	StatsWrite	length
numString	IntCallP	UsefmlCal	0	1	StatsWrite	WriteSCTitles	str			
z	IntCallP	Global	0	2	String	WriteRepeatInfo	z			
numString	IntCallP	UsefmlCal	0	1	String	WriteRepeatInfo	numString			
8	FindFileNumber	Term	fSP	FmlParam	-	0	0	-	-	-
			callLocation	FmlParam	-	0	0	-	-	-
			notFound	DataBase	Local	1	1	-	-	-
			callLocation	DataBase	UsefmlCal	0	2	-	-	-
			fSP	DataBase	UsefmlCal	0	2	-	-	-
			fileCount	DataBase	Local	1	3	-	-	-
			startIndex	DataBase	Dereferen	0	1	fSP	FindFileNumber	-
			endIndex	DataBase	Dereferen	0	1	fSP	FindFileNumber	-

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE			DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias
8	WriteTableTitles	nonT	fileCount	RetVal	Local	0	1	Stream	CheckReturnConditio	fileCount
			statsFile	IntCallp	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallp	Imported	0	1	Stream	WriteTableTitles	Ascii
			statsFile	IntCallp	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallp	Imported	0	1	Stream	WriteTableTitles	Ascii
			statsFile	IntCallp	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallp	Imported	0	1	Stream	WriteTableTitles	Ascii
			statsFile	IntCallp	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallp	Imported	0	1	Stream	WriteTableTitles	Ascii
			statsFile	IntCallp	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallp	Imported	0	1	Stream	WriteTableTitles	Ascii
			statsFile	IntCallp	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallp	Imported	0	1	Stream	WriteTableTitles	Ascii
			statsFile	IntCallp	External	0	1	Stream	WriteTableTitles	statsFile
			CR	IntCallp	Imported	0	1	Stream	WriteTableTitles	Ascii
9	AppendStringAndGrow	Term	str	FmlParam	-	0	0	-	-	-
			statsFile	IntCallp	External	0	1	Stream	StatsWrite	statsFile
			str	IntCallp	UseFmlCal	0	1	Stream	StatsWrite	str
			size	FmlParam	-	0	0	-	-	-
			length	FmlParam	-	0	0	-	-	-
9	PutChar	Term	size	Database	UseFmlCal	2	3	-	-	-
			length	Database	UseFmlCal	0	1	-	-	-
			numString	IntCallp	Global	0	1	String	AddSpaces	numString
			z	IntCallp	Global	0	1	String	AddSpaces	z
			size	FmlParam	-	0	0	-	-	-
9	ReduceLength	nonT	length	FmlParam	-	0	0	-	-	-
			subStringDescr	Database	Local	1	0	-	-	-
			numString	Database	Global	0	1	numString	StructChartImpl	-
			length	Database	UseFmlCal	0	1	-	-	-
			tempString	Database	Local	1	0	-	-	-
			z	IntCallp	Global	0	1	String	ReduceLength	z
			length	IntCallp	UseFmlCal	0	1	String	ReduceLength	length
			tempString	IntCallp	Local	0	1	String	ReduceLength	tempString
			subStringDescr	IntCallp	Local	0	1	String	ReduceLength	subStringDescr
			z	IntCallp	Global	0	1	String	ReduceLength	z
			numString	IntCallp	Global	1	1	String	ReduceLength	numString
			tempString	IntCallp	Local	0	1	String	ReduceLength	tempString
			z	IntCallp	Global	0	1	String	ReduceLength	z
			length	IntCallp	Global	0	1	String	ReduceLength	length
			tempString	IntCallp	Local	0	1	String	ReduceLength	tempString
			subStringDescr	IntCallp	Local	0	1	String	ReduceLength	subStringDescr

DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE

NODE				DATA				CONNECTION			
Lvl	Node Name	Type	Name	Type	Scope	Wr	Rd	Name	Declared in	Alias	
9	PutString	Term	z	IntCallP	Global	0	1	String	ReduceLength	z	
9	CheckLength	nonT	tempString	IntCallP	Local	0	1	String	ReduceLength	tempString	
			length	FmlParam	-	0	0	-	-	-	
			size	DataBase	Local	1	2	-	-	-	
			numString	IntCallP	Global	0	1	String	CheckLength	numString	
			length	DataBase	UseFmlCal	0	2	-	-	-	
			size	IntCallP	Local	0	1	AddSpaces	CheckLength	size	
			length	IntCallP	UseFmlCal	0	1	AddSpaces	CheckLength	length	
			size	IntCallP	Local	0	1	ReduceLength	AddSpaces	size	
			length	IntCallP	UseFmlCal	0	1	ReduceLength	AddSpaces	length	
10	PutString	Term									
10	AppendStringAndGrow	Term									
10	MakeString	Term									
10	AppendSubString	Term									
10	FreeString	Term									
10	CopyToNewString	Term									
10	Length	Term									
10	AddSpaces	nonT	size	FmlParam	-	0	0	-	-	-	
			length	FmlParam	-	0	0	-	-	-	
			size	DataBase	UseFmlCal	2	3	-	-	-	
			length	DataBase	UseFmlCal	0	1	-	-	-	
			numString	IntCallP	Global	0	1	String	AddSpaces	numString	
			z	IntCallP	Global	0	1	String	AddSpaces	z	
10	ReduceLength	nonT	size	FmlParam	-	0	0	-	-	-	
			length	FmlParam	-	0	0	-	-	-	
			numString	DataBase	Local	1	0	-	-	-	
			subStringDescr	DataBase	Global	0	1	numString	StructChartImpl	-	
			length	DataBase	UseFmlCal	0	1	-	-	-	
			tempString	DataBase	Local	1	0	-	-	-	
			z	IntCallP	Global	0	1	String	ReduceLength	z	
			length	IntCallP	Global	0	1	String	ReduceLength	length	
			tempString	IntCallP	UseFmlCal	0	1	String	ReduceLength	tempString	
			subStringDescr	IntCallP	Local	0	1	String	ReduceLength	subStringDescr	
			z	IntCallP	Local	0	1	String	ReduceLength	z	
			numString	IntCallP	Global	1	1	String	ReduceLength	numString	
			tempString	IntCallP	Local	0	1	String	ReduceLength	tempString	
			z	IntCallP	Global	0	1	String	ReduceLength	z	
			tempString	IntCallP	Local	0	1	String	ReduceLength	tempString	
			z	IntCallP	Global	0	1	String	ReduceLength	z	
			tempString	IntCallP	Local	0	1	String	ReduceLength	tempString	
11	AppendStringAndGrow	Term									
11	MakeString	Term									
11	AppendSubString	Term									
11	FreeString	Term									
11	CopyToNewString	Term									

APPENDIX B - SOURCE CODE LISTINGS

B.1 CONFIGURATION FILE

```
-- File: CMAAnalyzer.config - last edit:
-- Lewis:Wbst129:Xerox 25-Jun-87 10:27:46
-- Conley:Henr 4-Mar-89 19:12:31
-- Bonikowski:Henr801C:Xerox 3-Mar-89 18:07:04
-- Egerton:Wbst129 25-Nov-91 12:52:07
-- *****
-- File: DesignExtractor.config
-- Copyright (C) 1987-1992 by Xerox Corporation. All rights reserved.
-- This is the configuration for the automated mesa DesignExtractor tool.
-- *****
```

DesignExtractor: CONFIGURATION

```
IMPORTS Exec, Format, FormSW, Heap, HeraldWindow, MFile, MSegment,
MStream, Process, Put, Real,
Runtime, Stream, String, TextSW, Token, Tool,
ToolDriver, ToolWindow, UserInput
```

CONTROL DesignExtractorImpl = BEGIN

```
ChangeTableImpl;
CMFileImpl;
DataFlowImpl;
DesignExtractorImpl;
DisplayImpl;
Grammar;
MesaTab;
ParserImpl;
PassImpl;
ProgramParserImpl;
RawDataImpl;
StringTableImpl;
StructChartImpl;
UtilsImpl;
```

END. -- CONFIGURATION DesignExtractor

```
Compiler /jno ↑
ParseTable StringTable ↑
MyFormat Utils ↑
ChangeTable Parser Display ↑
UtilsImpl ChangeTableImpl CMFileImpl ↑
DataFlowImpl DisplayImpl StringTableImpl ↑
Grammar ParserImpl/j-bn RawDataImpl PassImpl ProgramParserImpl ↑
StructChartImpl DesignExtractorImpl ↑
```

Binder DesignExtractor

DesignExtractor

APPENDIX B - SOURCE CODE LISTINGS

B.2 INTERFACE FILES

```
Display.mesa      11-Nov-93 19:40:11 EST
```

```
-- File:Odisplay.mesa  Last edit:
-- Egerton:Wbst129:Xerox  30-Nov-91 13:10:13
--
--*****
-- File: Display.mesa
--
-- Copyright(C) 1991 by Xerox Corporation. All rights reserved.
--
-- Oavid Egerton:Wbst129
--
-- This module takes the data extracted by the DesignExtractor
-- and outputs the data to the file DesignExtractor.data in the
-- format selected at the window.
--
--*****
```

```
OIRECTORY
Parser;
```

```
Display: DEFINITIONS =
```

```
BEGIN
```

```
-- Exportable variable
-- Variables to maintain the total stack of procedures
-- calls and declarations
FullStackPtr: TYPE = LONG POINTER TO FullStack;
FullStack: TYPE = ARRAY[0..maxStack] OF FullStackProps;
FullStackProps: TYPE = RECORD [
    callLocation: CARDINAL ← 0,
    callLevel: CARDINAL ← 0,
    decilLocation: CARDINAL ← 0,
    fileNumber: CARDINAL ← 0,
    SLC: BOOLEAN ← FALSE,
    firstOfTwin: BOOLEAN ← FALSE,
    firstOfLevel: BOOLEAN ← FALSE,
    firstOfGlobal: BOOLEAN ← FALSE,
    twinRepeat: BOOLEAN ← FALSE,
    levelRepeat: BOOLEAN ← FALSE,
    globalRepeat: BOOLEAN ← FALSE,
    vSPIndex: CARDINAL ← 0];
```

```
maxStack: CARDINAL = Parser.MaxString/7;
```

```
-- From DisplayImpl
Control: PUBLIC PROCEDURE[created: BOOLEAN, aStackPtr: Parser.AtomStackPtr, aIndexPtr: Parser.AtomIndexPtr, extStackPtr: LONG POINTER, maxExternal:
CARDINAL, fileStackPtr: Parser.FileStackPtr, MaxFiles: CARDINAL, rawOdata:BOOLEAN, SCGraph:BOOLEAN, SCTable:BOOLEAN, OFOVariate:BOOLEAN,
OFDTable:BOOLEAN, debugg:BOOLEAN, repeats:BOOLEAN] RETURNS[BOOLEAN];
```

```
-- From RawDataImpl
RawOdata: PUBLIC PROCEDURE[created: BOOLEAN, aStackPtr: Parser.AtomStackPtr,
aIndexPtr: Parser.AtomIndexPtr, fileStackPtr: Parser.FileStackPtr,
MaxFiles: CARDINAL, debugg:BOOLEAN] RETURNS[BOOLEAN];
```

```
Odisplay.mesa      11-Nov-93 19:40:11 EST
```

```

-- From StructChartImpl
StructChart: PUBLIC PROCEDURE[created: BOOLEAN, aStackPtr: Parser.AtomStackPtr,
aIndexPtr: Parser.AtomIndexPtr, extSP: LONG POINTER, maxExt: CARDINAL,
fileStackPtr: Parser.FileStackPtr, MaxFiles: CARDINAL,
SCGraph: BOOLEAN, SCTable: BOOLEAN, DFovVariable: BOOLEAN, DFOTable: BOOLEAN,
debugg: BOOLEAN, repeats: BOOLEAN] RETURNS[BOOLEAN];

-- From ImpExpImpl
-- ImpExp: PUBLIC PROCEDURE[created: BOOLEAN, aStackPtr: Parser.AtomStackPtr,
-- aIndexPtr: Parser.AtomIndexPtr, fileStackPtr: Parser.FileStackPtr,
-- MaxFiles: CARDINAL, debugg: BOOLEAN] RETURNS[BOOLEAN];

-- From DataFlowImpl
DataFlow: PUBLIC PROCEDURE[created: BOOLEAN, aStackPtr: Parser.AtomStackPtr, aIndexPtr:
Parser.AtomIndexPtr, fileStackPtr: Parser.FileStackPtr, MaxFiles: CARDINAL,
OFDVariable: BOOLEAN, OFOTable: BOOLEAN, debugg: BOOLEAN]
RETURNS[BOOLEAN];

-- From DataFlowImpl
DataFlow: PUBLIC PROCEDURE[aIP: Parser.AtomIndexPtr, aSP: Parser.AtomStackPtr,
extSP: LONG POINTER, maxExt: CARDINAL, fSP: Parser.FileStackPtr, MaxFiles: CARDINAL,
positionStackPtr: LONG POINTER, fullStackPtr: LONG POINTER, fullCount: CARDINAL,
OFovVariable: BOOLEAN, OFOTable: BOOLEAN, debugg: BOOLEAN];

-- From DataDictImpl
DataDict: PUBLIC PROCEDURE[created: BOOLEAN, aStackPtr: Parser.AtomStackPtr,
aIndexPtr: Parser.AtomIndexPtr, fileStackPtr: Parser.FileStackPtr,
MaxFiles: CARDINAL, debugg: BOOLEAN] RETURNS[BOOLEAN];

-- From CommentImpl
Comments: PUBLIC PROCEDURE[created: BOOLEAN, aStackPtr: Parser.AtomStackPtr,
aIndexPtr: Parser.AtomIndexPtr, fileStackPtr: Parser.FileStackPtr,
MaxFiles: CARDINAL, debugg: BOOLEAN] RETURNS[BOOLEAN];
ENO.

```



```
Parser.mesa      13-Nov-93 10:02:32 EST
```

```
-- File: Parser.mesa - last edit:
-- Bonikowski:Henr801C:Xerox 25-Apr-89 12:21:17
-- Conley.Henr      4-Mar-89 18:59:38
-- Egerton:Wbst129  13-Nov-93 09:16:22

-- *****
--
-- File: Parser.mesa
-- Copyright (C) 1989 - 1992 by Xerox Corporation. All rights reserved.
--
-- David Egerton:Wbst129:Xerox
--
-- This module contains the interface for various routines and data structures
-- used in the DesignExtractor Program analyzer Modified from P1.mesa in the
-- in the Mesa compiler Also the programs in the DesignExtractor parsing
-- sequence are defined here.
-- *****
--
OIRECTORY

MStream USING [Handle],
ParseTable: TYPE USING [ActionEntry, ProdDataRef, State, TableRef, TSymbol],
Stream: TYPE USING [Handle];

Parser: DEFINITIONS =
BEGIN

OPEN ParseTable;

ValueStack: TYPE = LONG POINTER TO ValueSeq;
StateStack: TYPE = LONG POINTER TO StateSeq;
LinkStack: TYPE = LONG POINTER TO LinkSeq;
ActionStack: TYPE = LONG POINTER TO ActionSeq;

Value: TYPE = LONG STRING; --type of the elements of the value stack

ValueSeq: TYPE = RECORD [SEQUENCE length: NAT OF Value];
StateSeq: TYPE = RECORD [SEQUENCE length: NAT OF State];
LinkSeq: TYPE = RECORD [SEQUENCE length: NAT OF CAROINAL];
ActionSeq: TYPE = RECORD [SEQUENCE length: NAT OF ActionEntry];

Token: TYPE = RECORD [
  class: TSymbol,      -- token class
  value: LONG STRING,  -- token value
  index: CAROINAL];    -- source line index

maxIds: CAROINAL = 50;
idsList: TYPE = ARRAY [1..maxIds] OF LONG STRING;

ids: idslst;
idindex: CAROINAL;

-- Variables for the file level atomStack DataBase
FileStackPtr: TYPE = LONG POINTER TO FileStack;
FileStack: TYPE = ARRAY[0..MaxFiles] OF FileProps;

Parser.mesa      13-Nov-93 10:02:32 EST
```

```

FileProps: TYPE = RECORD [
  fileName: LONG STRING ← NIL,
  fileCount: CARDINAL ← 0,
  exportName: LONG STRING ← NIL,
  startIndex: CARDINAL ← 0,
  endIndex: CARDINAL ← 0,
  procCount: CARDINAL ← 0,
  prevLevel: CARDINAL ← 0];
MaxFiles: CARDINAL = 30;

AtomStackPtr: TYPE = LONG POINTER TO AtomStack;
AtomStack: TYPE = RECORD[SEQUENCE length: NAT OF AtomProps];
AtomProps: TYPE = RECORD [
  BElevel: CARDINAL ← 0,
  declLocation: CARDINAL ← 0,
  fileNumber: CARDINAL,
  fileType: {Program,Monitor,Configuration,Definitions},
  idMark: CARDINAL ← 0,
  IEatom: LONG STRING ← NIL,
  IEatom: CHAR,
  Index: CARDINAL ← 0,
  inputString: LONG STRING ← NIL,
  listMark: CARDINAL ← 0,
  plevel: CARDINAL ← 0,
  readWrite: CHAR,
  ruleValue: CARDINAL,
  sCount: CARDINAL ← 0,
  symbolNumber: CARDINAL ← 0,
  type: LONG STRING ← NIL,
  typeBase: LONG STRING ← NIL,
  varMark: CARDINAL ← 0,
  varScope: CHAR,
  values: LONG STRING ← NIL];

MaxString: CARDINAL = 40001;

AtomIndexPtr: TYPE = LONG POINTER TO AtomIndex;
AtomIndex: TYPE = CARDINAL;

-- From Pass1Impl.mesa
-- The pass1 parsing function achieves the following objectives:
--   - Identifies each Mesa keyword
--   - Identifies all non Mesa identifiers
--   - Creates the original atom stack database

Atom: PROC [] RETURNS [Token]:
BaseTable: PUBLIC PROC [
  inputSymbol: Value, symbolNumber: ParseTable.TSymbol,
  aIP: AtomIndexPtr, aSP: AtomStackPtr, fileCount: CARDINAL, ruleValue: CARDINAL]
  RETURNS [atomSP: LONG POINTER, uniqueAtom: BOOLEAN];
ErrorContext: PROC [to: Stream.Handle, message: LONG STRING,
  tokenIndex: CARDINAL];
ResetScanIndex: PROC [CARDINAL] RETURNS [success: BOOLEAN];
ScanInit: PROC [table: ParseTable.TableRef, infile: MStream.Handle,
  z: UNCOUNTED ZONE];
ScanReset: PROC;

```

```
ScanStats: PROC RETURNS [nLines, nErrors: CAROINAL];
```

```
-- From ParserImpl.mesa
```

```
Parse: PUBLIC PROCEDURE [cz: UNCOUNTED_ZONE, aStackPtr: Parser.AtomStackPtr, aIndexPtr: Parser.AtomIndexPtr, fileStackPtr: Parser.FileStackPtr,
fileCount:CAROINAL, rawData:BOOLEAN] RETURNS[atomSP:LONG POINTER];
```

```
ParserIO: PROC RETURNS [LONG STRING];
```

```
InputLoc: PROC RETURNS [CAROINAL];
```

```
-- From Grammar.mesa
```

```
-- Grammar is not used in the Oesign Extractor program although the
-- rules and productions are maintained for completeness. The rule
-- and production values could be used at some later date should
-- they be required.
```

```
AssignDescriptors: PROC [
qd: ActionStack, vd: ValueStack, ld: LinkStack, pp: ParseTable.ProdDataRef,
z: UNCOUNTED_ZONE];
```

```
ProcessQueue: PROC [qI, top: CAROINAL] RETURNS[rule:CAROINAL];
```

```
TokenValue: PROC [ISymbol] RETURNS [Value];
```

```
-- From ProgramParserImpl.mesa
```

```
-- The program parser is responsible for parsing Program and Monitor
-- files. Pass 2 is responsible for identifying contexts:
```

```
-- - Mark all begin and end blocks
-- - Mark all statement boundaries
-- - Mark all procedure boundaries
-- - Mark all lists, ie Directory, Imports, Exports, Parameters,
--   Return paramaters, call lists & others.
-- Pass 3 is responsible for resolving all non Mesa identifiers & var types:
-- - Program names
-- - Procedure names
-- - Variable names
-- - Variable types
-- - Variable read/write usage
-- - Variable Import locations
-- - Variable declarations
```

```
-- Pass 4 is responsible for counting the procedure declaration
-- Pass 5 is responsible for determining the scope of each variable
```

```
-- - L = Local
--   G = Global
--   P = Public
```

```
Pass2: PUBLIC PROC [
inputSymbol:Value, symbolNumber:ParseTable.ISymbol,aIP:AtomIndexPtr,
```

```

    aSP: AtomStackPtr, fileStackPtr:Parser.FileStackPtr, fileCount:CARDINAL,
    ruleValue:CARDINAL]
    RETURNS[uniqueAtom:BOOLEAN];

Pass3: PUBLIC PROC [
    inputSymbol:Value, symbolNumber:ParseTable.TSymbol, aIP:AtomIndexPtr,
    aSP: AtomStackPtr, fileStackPtr:Parser.FileStackPtr, fileCount:CARDINAL,
    ruleValue:CARDINAL]
    RETURNS[uniqueAtom:BOOLEAN];

Pass4: PUBLIC PROC [aIP:AtomIndexPtr, aSP: AtomStackPtr, fSP:Parser.FileStackPtr,
    fileCount:CARDINAL];

Pass5: PUBLIC PROC [aIP:AtomIndexPtr, aSP: AtomStackPtr, fSP:Parser.FileStackPtr,
    fileCount:CARDINAL];

END. -- DEFINITIONS Parser

```

```

-- File: ProcMetrics.mesa - last edit:
-- Conley, Henry 10-Mar-89 20:07:31
-- *****
--
-- File: ProcMetrics.mesa
--
-- Copyright (C) 1989 by Xerox Corporation. All rights reserved.
--
-- Lewis:Wbst129:Xerox
-- 19-Jun-87 8:14:21
--
-- This module contains the interface for the procedural metric collection
-- routines and data structures.
--
-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.
-- *****
--
DIRECTORY ;

ProcMetrics: DEFINITIONS =
BEGIN

    PmMaxNest: CARDINAL = 20;

    PmLink: TYPE = LONG POINTER TO PmNode;

    PmNode: TYPE = RECORD [
        pmName: LONG STRING,
        pmHalstead: HsHeadLink,
        pmMcCabe: McLink,
        pmNumLines: CARDINAL];

    PmStackArray: TYPE = ARRAY [0..PmMaxNest] OF PmLink;

    -- Halstead collection structures

    HsLink: TYPE = LONG POINTER TO HsNode;

    HsHeadLink: TYPE = LONG POINTER TO HsHead;

    HsNode: TYPE = RECORD [
        hsId: LONG STRING,
        hsCount: CARDINAL,
        hsNext: HsLink];

    HsHead: TYPE = RECORD [
        hsOperators,
        hsOperands: HsLink];

    -- McCabe collection structure

    mcIf:
    mcSelectOption:
        CARDINAL = 1;
        CARDINAL = 2;

ProcMetrics.mesa 10-Mar-89 20:07:31 EST

```

```
mcWhile:    CARDINAL = 3;
mcUntil:    CARDINAL = 4;
mcThrough:  CARDINAL = 5;
mcFor:      CARDINAL = 6;
mcAnd:      CARDINAL = 7;
mcOr:       CARDINAL = 8;
mcCatch:    CARDINAL = 9;
mcFork:     CARDINAL = 10;

mcMax:      CARDINAL = 10;

McLink:     TYPE = LONG POINTER TO McCounts;

McCounts:   TYPE = ARRAY [1..mcMax] OF CARDINAL;

-- from ProcMetricsImpl.mesa

PmInit: PROCEDURE;

PmOperator: PROCEDURE [pm: PmLink, pmOpName: LONG STRING];

PmOperand: PROCEDURE [pm: PmLink, pmOpName: LONG STRING];

PmMcCabe: PROCEDURE [pm: PmLink, mcTerm: CARDINAL];

PmLines: PROCEDURE [pm: PmLink];

PmPush: PROCEDURE [pm: PmLink, procName: LONG STRING]
        RETURNS [PmLink];

PmPop: PROCEDURE [pm: PmLink] RETURNS [PmLink];

PmImpress: PROCEDURE [pm: PmLink];

PmDump: PROCEDURE [pm: PmLink];

-- from HalsteadImpl.mesa

HsInit: PROCEDURE RETURNS [HsHeadLink];

HsOperator: PROCEDURE [hs: HsHeadLink, hsOpName: LONG STRING];

HsOperand: PROCEDURE [hs: HsHeadLink, hsOpName: LONG STRING];

HsCompute: PROCEDURE [hs: HsHeadLink] RETURNS [CARDINAL, REAL, REAL];

HsDump: PROCEDURE [hs: HsHeadLink];

-- from McCabeImpl.mesa
```

```
McInit: PROCEDURE RETURNS [McLink];

McIncrement: PROCEDURE [mcc: McLink, mcTerm: CARDINAL];

McCompute: PROCEDURE [mcc: McLink] RETURNS [CARDINAL];

McDump: PROCEDURE [mcc: McLink];

-- From StatsImpl.mesa

MetricStats: PROCEDURE;

END. -- DEFINITIONS ProcMetrics

Conley 10-Mar-89: Added PmInit
```

```
-- File: Utils.mesa - last edit:
-- Bonikowski:Henr801C:Xerox 28-Apr-89 10:32:12
-- *****
-- File: Utils.mesa
--
-- Copyright (C) 1989 by Xerox Corporation. All rights reserved.
--
-- David Egerton:Wbst129
--
-- This module contains the interface for several utility routines and data
-- structures used in the DesignExtractor. Information from John Lewis
-- program of 1987 was adopted for use here.
--
-- Copyright (C) 1987 to 1992 by Xerox Corporation. All rights reserved.
-- *****
OIRECTORY
  Format USING [StringProc],
  Stream USING [Position],
  Window USING [Handle];
Utils: DEFINITIONS =
  BEGIN
    IndexType: TYPE = CHARACTER ['A..'Z'];
    TableType: TYPE = ARRAY IndexType OF LONG STRING;
    maxNameLength: CAROINAL = 100;
    NBody: TYPE = PACKED ARRAY [1..maxNameLength] OF CHARACTER;
    MetRec: TYPE = RECORDO [
      loc, cc, N: CAROINAL, V, effort: REAL, nameLength: CAROINAL, nameBody: NBody];
    MetPoint: TYPE = LONG POINTER TO MetRec;
    MetricRec: TYPE = RECORDO [
      name: LONG STRING,
      loc, cc, N: CAROINAL,
      V, effort: REAL,
      used: BOOLEAN,
      next: MetricPoint];
    MetricPoint: TYPE = LONG POINTER TO MetricRec;
    StatArray: TYPE = ARRAY [1..3] OF REAL;
    StatsRec: TYPE = RECORDO [avg: StatArray, stdev: StatArray, count: CAROINAL];
    SortSeq: TYPE = LONG POINTER TO SortRec;
    SortRec: TYPE = RECORDO [
      count: CAROINAL, item: SEQUENCE size: [1..LAST[CAROINAL]] OF MetricPoint];
```



```

-- From AnalyzerImpl.mesa

window, formSW, fileSW: Window.Handle;

permZone, tempZone: UNCOUNTED_ZONE;
-- permZone is is not destroyed until the tool is unloaded.
-- tempZone may be destroyed after each file is processed.

backend: BOOLEAN;
encode: BOOLEAN;

overlap: CARDINAL;

log: Format.StringProc; -- use this to write info to the log.

CheckForAbort: PROC;
-- Raises ERROR ABORTED if stop key has been pressed.

-- From UtilsImpl.mesa

StringEqual: PROCEDURE [str1, str2: LONG_STRING] RETURNS [BOOLEAN];

StringLess: PROCEDURE [str1, str2: LONG_STRING] RETURNS [BOOLEAN];

CardToSString: PROCEDURE [num: CARDINAL] RETURNS [LONG_STRING];

RealToSString: PUBLIC PROCEDURE [num: REAL] RETURNS [LONG_STRING];

-- From CMFileImpl.mesa

FileClosingOptions: TYPE = {currentPos, fileLength, pos};

OpenMetFile: PROCEDURE;

CloseMetFile: PROCEDURE [
  truncateTo: FileClosingOptions + currentPos,
  pos: Stream.Position + 0];
-- truncateTo:
--   currentPos = current stream position
--   fileLength = don't truncate (don't change file length)
--   pos = set position to argument 'pos'

ResetMetFile: PROCEDURE;

MetFileLength: PUBLIC PROC RETURNS [Stream.Position];

EOFMetFile: PROCEDURE RETURNS [BOOLEAN];

StoreMetRec: PROCEDURE [inrec: MetRec];

GetMetRec: PROCEDURE [z: UNCOUNTED_ZONE] RETURNS [outNode: MetPoint];

ClearMetFile: PROCEDURE;

```

```
-- From EncodeImpl.mesa
```

```
EncodeId: PROCEDURE [idString: LONG STRING] RETURNS [LONG STRING];
```

```
DecodeId: PROCEDURE [idString: LONG STRING] RETURNS [LONG STRING];
```

```
ENO. -- DEFINITIONS Utils
```

```
Change Log
```

```
-----
```

```
Bonikowski 03/03/89 Added "log"
```

APPENDIX B - SOURCE CODE LISTINGS

B.3 PROGRAM FILES

```
-- EgertonWbst129:Xerox 11-Nov-93 17:17:28
-- *****
-- File: DataFlowImpl1.mesa
--
-- David Egerton:Wbst129
--
-- Copyright (C) 1989 by Xerox Corporation. All rights reserved.
--
-- This module is responsible for the collection of data into
-- the output format and writing to the file DataFlow.data.data
-- *****
```

DIRECTORY

```
  Ascii,
  Display,
  Environment,
  Heap,
  MFile,
  MStream,
  Stream,
  String,
  Parser,
  Utils;
```

DataFlowImpl1: PROGRAM

```
  IMPORTS Heap, MStream, Stream, String, Utils
  EXPORTS Display =
```

BEGIN

```
-- Variables to maintain the stack
-- of variables as they relate to each
VarStackPtr: TYPE = LONG POINTER TO VarStack;
VarStack: TYPE = ARRAY[0..maxVarStack] OF VarStackProps;
VarStackProps: TYPE = RECORD [
  index: CARDINAL + 0,
  readCount: CARDINAL + 0,
  writeCount: CARDINAL + 0];
maxVarStack: CARDINAL = Parser.MaxString/10;
varCount: CARDINAL + 0;

-- Variables to maintain the stack
-- of DataFlow sequence information
DFStackPtr: TYPE = LONG POINTER TO DFStack;
DFStack: TYPE = ARRAY[0..maxDFStack] OF DFStackProps;
DFStackProps: TYPE = RECORD [
  context: BOOLEAN + FALSE,
  SLC: BOOLEAN + FALSE,
  level: CARDINAL + 0,
  nodeName: CARDINAL + 0,
  nodeType: {Dummy,nonT,Term},
  nextIndex: CARDINAL + 0,

  maxAtomLength: CARDINAL + 20;
  maxBEndLength: CARDINAL + 4;
  maxFileLength: CARDINAL + 25;
  maxIDmarkLength: CARDINAL + 4;
  maxIEidentLength: CARDINAL + 15;
  maxIEoplLength: CARDINAL + 2;
  maxIndexLength: CARDINAL + 6;
  maxLevelLength: CARDINAL + 4;
  maxListMarkLength: CARDINAL + 4;
  maxLVLlength: CARDINAL + 3;
  maxProcLevelLength: CARDINAL + 4;
  maxProcLength: CARDINAL + 20;
  maxRepeatLength: CARDINAL + 4;
  maxRuleNumberLength: CARDINAL + 4;
  maxStmntLength: CARDINAL + 4;
  maxStringLength: CARDINAL + 40;
  maxSubMargin: CARDINAL + 74;
  maxSymbolNumberLength: CARDINAL + 4;
  maxTypeLength: CARDINAL + 15;
  maxTypeBaseLength: CARDINAL + 9;
  maxVarMarkLength: CARDINAL + 4;
  maxVarScopeLength: CARDINAL + 4;
  maxValuesLength: CARDINAL + 25;

  maxDFStack: CARDINAL = 1000;
  DFCount: CARDINAL + 0;

  -- Miscellaneous variables
  created: BOOLEAN + TRUE;
  debugg: BOOLEAN + FALSE;
  result: BOOLEAN + FALSE;
  pageSize: CARDINAL + 58;

  -- Formatting variables
  maxAtomLength: CARDINAL + 20;
  maxBEndLength: CARDINAL + 4;
  maxFileLength: CARDINAL + 25;
  maxIDmarkLength: CARDINAL + 4;
  maxIEidentLength: CARDINAL + 15;
  maxIEoplLength: CARDINAL + 2;
  maxIndexLength: CARDINAL + 6;
  maxLevelLength: CARDINAL + 4;
  maxListMarkLength: CARDINAL + 4;
  maxLVLlength: CARDINAL + 3;
  maxProcLevelLength: CARDINAL + 4;
  maxProcLength: CARDINAL + 20;
  maxRepeatLength: CARDINAL + 4;
  maxRuleNumberLength: CARDINAL + 4;
  maxStmntLength: CARDINAL + 4;
  maxStringLength: CARDINAL + 40;
  maxSubMargin: CARDINAL + 74;
  maxSymbolNumberLength: CARDINAL + 4;
  maxTypeLength: CARDINAL + 15;
  maxTypeBaseLength: CARDINAL + 9;
  maxVarMarkLength: CARDINAL + 4;
  maxVarScopeLength: CARDINAL + 4;
  maxValuesLength: CARDINAL + 25;

  -- Lexical variables
  assignment: CARDINAL + 209;
  callParameter: CARDINAL + 238;
  comma: CARDINAL + 9;
  commentString: CARDINAL + 6;
  equals: CARDINAL + 15;
  listMark: CARDINAL + 201;
  monitor: CARDINAL + 44;
  openBracket: CARDINAL + 145;
  program: CARDINAL + 43;
  parameterCall: CARDINAL + 238;
  parameter: CARDINAL + 200;
  point: CARDINAL + 27;
  procCall: CARDINAL + 237;
  procedure: CARDINAL + 37;
```



```

ELSE Utils.log["Updating DataBase with DataFlow Info"];

Stream.PutChar [statsFile, Ascii.CR];
lineCount ← lineCount + 1;

-- Collect variable information into variable stack
-- Take each file at a time
startIndex ← fSP[fileCount].startIndex;
WHILE posIndex < aIP DO
    posIndex ← posIndex + 1;

    SELECT aSP[posIndex].idMark FROM

    program => BEGIN
        varCount ← varCount + 1;
        vSP[varCount].index ← aSP[posIndex].Index;
    END;

    monitor => BEGIN
        varCount ← varCount + 1;
        vSP[varCount].index ← aSP[posIndex].Index;
    END;

    procedure => BEGIN
        varCount ← varCount + 1;
        vSP[varCount].index ← aSP[posIndex].Index;
        UpdateFullStackIndex[fullStackPtr, aSP, fullCount
        ];
    END;

    variable => BEGIN
        CheckVariableStatus[aIP, aSP, levelCount,
        posIndex, startIndex];
    END;

ENDCASE;

IF posIndex = fSP[fileCount].endIndex THEN
    {
        fileCount ← fileCount + 1;
        startIndex ← fSP[fileCount].startIndex;
        levelCount ← 0;
    };
ENDLOOP; -- posIndex

IF DFOVariable THEN OutputVariableInfo[aIP, aSP, fSP];

IF DFDTable THEN
    {
        OutputDFTitles[];

        ConstructContextDiagram[aIP, aSP, extSP, maxExt, fSP,
        MaxFiles, fullStackPtr, fullCount];

        ConstructLevelZero[aIP, aSP, extSP, maxExt, fSP, MaxFiles,
        fullStackPtr, fullCount];
    }

```

```

ConstructLevelDiagrams[aIP, aSP, extSP, maxExt, fSP, MaxFiles,
fullStackPtr, fullCount];

Utils.log[" - Writing dataflow table.....\n"];
};

varCount ← 0;
z.FREE[@vSP];

END; -- ProcessDataFlow

-- UpdateFullStackIndex: This procedure correlates the vSP
(variable
-- database) with the fullStack database. This enables quick ref
-- of the sort of variables against each procedure, monitor and
-- program
UpdateFullStackIndex: PROCEDURE
[fullStackPtr:Display.FullStackPtr, aSP:Parser.AtomStackPtr,
fullCount:CARDINAL] =
BEGIN
    --Local Variables
    found: BOOLEAN ← FALSE;
    count: CARDINAL ← 0;
    declLocation: CARDINAL ← 0;
    index: CARDINAL ← 0;

    WHILE count < fullCount DO
        count ← count + 1;

        declLocation ← vSP[varCount].index;
        IF declLocation = fullStackPtr[count].declLocation THEN
            {
                fullStackPtr[count].vSPIndex ← varCount;
            };
        ENDLOOP;

    END; -- UpdateFullStackIndex

-- Writes the file information on the first page
WriteFiles: PROCEDURE [ fileStackPtr: Parser.FileStackPtr, MaxFiles:
CARDINAL] =
BEGIN
    -- Local variables
    maxFileIndexLength: CARDINAL ← 5;
    maxFileNameLength: CARDINAL ← 30;
    maxStartLength: CARDINAL ← 11;
    maxEndLength: CARDINAL ← 9;

```

```
maxExportNameLength: CARDINAL ← 25;
```

```
-- Procedure starts here
```

```
tempIndex: CARDINAL ← 0;
```

```
titleSize: CARDINAL ← 6;
```

```
StatsWrite ["DESIGN EXTRACTION STATISTICS" L];
```

```
Stream.PutChar [statsFile, Ascii.CR];
```

```
StatsWrite ["===== " L];
```

```
Stream.PutChar [statsFile, Ascii.CR];
```

```
Stream.PutChar [statsFile, Ascii.CR];
```

```
StatsWrite ["The files parsed in this session were: " L];
```

```
Stream.PutChar [statsFile, Ascii.CR];
```

```
Stream.PutChar [statsFile, Ascii.CR];
```

```
StatsWrite [" Index File Name
```

```
Index End Index Exports to " L];
```

```
Stream.PutChar [statsFile, Ascii.CR];
```

```
StatsWrite ["----- " L];
```

```
Stream.PutChar [statsFile, Ascii.CR];
```

```
-- Update the line count
```

```
lineCount ← lineCount + titleSize;
```

```
-- Output file data
```

```
WHILE tempIndex < MaxFiles DO
```

```
tempIndex ← tempIndex + 1;
```

```
lineCount ← lineCount + 1;
```

```
IF fileStackPtr[tempIndex].fileCount = 0 THEN EXIT;
```

```
-- Write out file count
```

```
StatsWrite [" " L];
```

```
numString ←
```

```
String.CopyToString[Utils.CardToString[fileStackPtr[tempIn
```

```
dex].fileCount], z];
```

```
CheckLength[maxFileIndexLength];
```

```
StatsWrite [numString];
```

```
String.FreeString [z, numString];
```

```
-- Write out file name
```

```
StatsWrite [" " L];
```

```
numString ←
```

```
String.CopyToString[fileStackPtr[tempIndex].fileName, z];
```

```
CheckLength[maxFileNameLength];
```

```
StatsWrite [numString];
```

```
String.FreeString [z, numString];
```

```
-- Write out file start index
```

```
StatsWrite [" " L];
```

```
numString ←
```

```
String.CopyToString[Utils.CardToString[fileStackPtr[tempIn
```

```
dex].startIndex], z];
```

```
CheckLength[maxStartLength];
```

```
StatsWrite [numString];
```

```
String.FreeString [z, numString];
```

```
-- Write out file end index
```

```
StatsWrite [" " L];
```

```
numString ←
```

```
String.CopyToString[Utils.CardToString[fileStackPtr[tempIn
```

```
dex].endIndex], z];
```

```
CheckLength[maxEndLength];
```

```
StatsWrite [numString];
```

```
String.FreeString [z, numString];
```

```
-- Write out export information
```

```
StatsWrite [" " L];
```

```
numString ←
```

```
String.CopyToString[fileStackPtr[tempIndex].exportName, z];
```

```
CheckLength[maxExportNameLength];
```

```
StatsWrite [numString];
```

```
String.FreeString [z, numString];
```

```
Stream.PutChar [statsFile, Ascii.CR];
```

```
ENDLOOP; -- File output
```

```
-- Complete page
```

```
WHILE lineCount MOD pageSize # 0 DO
```

```
lineCount ← lineCount + 1;
```

```
Stream.PutChar [statsFile, Ascii.CR];
```

```
ENDLOOP; -- Complete Page
```

```
END; -- WriteFiles
```

```
-- Writes a string to the statistics output file
```

```
StatsWrite: PROCEDURE [str: LONG STRING] =
```

```
BEGIN
```

```
Stream.PutString [statsFile, str];
```

```
END; -- StatsWrite
```

```
-- Checks the length of the string and adds in spaces if short
```

```
CheckLength: PROCEDURE [length: CARDINAL] =
```

```
BEGIN
```

```
-- Local variable
```

```
size: CARDINAL;
```

```
-- Procedure body
```

```
size ← String.Length[numString];
```

```
IF size < length THEN AddSpaces[size, length];
```

```
IF size > length THEN ReduceLength[size, length];
```

```
END; -- CheckLength;
```

```
AddSpaces: PROCEDURE [size: CARDINAL, length: CARDINAL] =
```

```

BEGIN
  size ← length - size;  -- Determine how many spaces need to be
  added
  WHILE size > 0 DO
    String.AppendStringAndGrow[@numString, ' ', z];
    size ← size - 1;
  ENDLOOP;
END;  -- AddSpaces

ReduceLength: PROCEDURE[size: CARDINAL, length: CARDINAL] =
BEGIN
  -- Substring allows the extraction of a portion of a string
  subStringDescr: String.SubStringDescriptor;
  tempString: LONG STRING ← NIL;

  -- Procedure body
  subStringDescr ← [numString.0, length];
  tempString ← String.MakeString[z, length];

  String.AppendSubString[tempString, @subStringDescr];
  String.FreeString[z, numString];
  numString ← String.CopyToNewString[tempString, z];
  String.FreeString[z, tempString];

END;  -- ReduceLength

SendProgressMessage: PROCEDURE [ti: CARDINAL] =
BEGIN
  count: LONG STRING;

  IF ti = 0 THEN Utils.log["\nLines processed ... "L];

  IF ti MOD 100 = 0 THEN {
    count ← Utils.CardToCString[ti];
    Utils.log[count];
    Utils.log[" ", "L"];
  };

END;  -- SendProgressMessage

-- CheckVariableStatus: This procedure decorates the varStack with
information
-- relevant to variables that are written too or read from.
CheckVariableStatus: PROCEDURE [aIP: LONG POINTER, aSP:
  Parser.AtomStackPtr,
    levelCount: CARDINAL,
    posIndex: CARDINAL,
]

```

```

startIndex: CARDINAL] =
BEGIN
  -- Local variable declarations
  backUpIndex: CARDINAL ← posIndex;
  currentLevel: CARDINAL ← 0;
  found: BOOLEAN ← FALSE;
  tempVarCount: CARDINAL ← 0;

  -- Procedure starts here

  -- Check to see if this variable is already in the database
  -- at the procedure level currently under test
  tempVarCount ← varCount + 1;
  currentLevel ← aSP[posIndex].varMark;
  -- double counting dereferenced variables where the
  -- parent and the child are both marked variable
  -- Also bypass return values which will already have
  -- been used but need to be registered separately.
  IF aSP[posIndex].listMark # return THEN
  {
    WHILE tempVarCount > 0 DO
      tempVarCount ← tempVarCount - 1;
      IF aSP[vSP[tempVarCount].index].varMark < currentLevel
      THEN EXIT;
      IF currentLevel = 0 THEN IF
        aSP[vSP[tempVarCount].index].varMark > 0 THEN EXIT;
      IF aSP[posIndex].varScope # 'M AND aSP[posIndex].varScope
        # 'N THEN
      {
        -- Look for a declaration match for non imported
        variables
        IF aSP[posIndex].declLocation =
          aSP[vSP[tempVarCount].index].declLocation THEN
        {
          -- Inhibit count if the variable is an internal
          call parameter
          -- These are required for display later
          IF aSP[posIndex].listMark # callParameter THEN
          {
            -- Update the count against the current
            variable
            IF aSP[posIndex].readWrite = 'W THEN
            {
              vSP[tempVarCount].writeCount ←
              vSP[tempVarCount].writeCount + 1;
            }
            ELSE
            {
              vSP[tempVarCount].readCount ←
              vSP[tempVarCount].readCount + 1;
            };
            found ← TRUE;
            EXIT;
          }
          ELSE EXIT;
        }
      }
    }
  }

```



```

ELSE
{
  If String.Equal[aSP[posIndex].inputString,
  aSP[vSP[tempVarCount].index].inputString] THEN
  {
    -- Inhibit count if the variable is an internal
    call parameter
    -- These are required for display later
    IF aSP[posIndex].listMark # callParameter THEN
    {
      -- Update the count against the current
      imported variable
      IF aSP[posIndex].readWrite = 'W' THEN
      {
        vSP[tempVarCount].writeCount ←
        vSP[tempVarCount].writeCount + 1;
      }
      ELSE
      {
        vSP[tempVarCount].readCount ←
        vSP[tempVarCount].readCount + 1;
      }
      found ← TRUE;
      EXIT;
    }
    ELSE EXIT;
  }
};

ENDLOOP; -- varCount
};

-- If not found put a new entry in the database
IF NOT found THEN
{
  varCount ← varCount + 1;
  IF varCount >= (maxVarStack) THEN {Utils.log["\nVariable
count overflow error.....\n"]};
  vSP[varCount].index ← aSP[posIndex].Index;
  vSP[varCount].writeCount ← vSP[varCount].writeCount +
  1;
}
ELSE
{
  vSP[varCount].readCount ← vSP[varCount].readCount +
  1;
};
};

END; -- CheckVariableStatus

```

```

-- OutputVariable information: This procedure outputs the variable
information
-- parsed from the main database and the variable stack previously
created. The
-- Output is in sequential order of procedure declaration.
OutputVariableInfo: PROCEDURE[aIP: LONG POINTER, aSP:
Parser.AtomStackPtr, fSP: Parser.FileStackPtr] =
BEGIN
  -- Local variables
  foundProcCall:BOOLEAN ← FALSE;
  index: CARDINAL ← 0;
  tempVarCount: CARDINAL ← 1;
  tempIndex: CARDINAL ← 0;
  fileCount: CARDINAL ← 0;
  levelCount: CARDINAL ← 0;
  startingIndex:CARDINAL ← 0;
  offset: CARDINAL ← 0;

  -- Local size information
  maxName: CARDINAL ← 20;
  maxField: CARDINAL ← 5;
  maxType: CARDINAL ← 11;

  -- Procedure to take care of page endings
  CompletePage: PROCEDURE =
  BEGIN
    WHILE lineCount MOD pageSize # 0 DO
      lineCount ← lineCount + 1;
      Stream.PutChar [statsFile, Ascii.CR];
    ENDLOOP; -- Complete Page
    END; -- CompletePage

  -- Procedure to check for page breaks
  CheckIfPageBreak: PROCEDURE =
  BEGIN
    IF lineCount MOD pageSize = 0 THEN
      OutputVarTableTitles[];
    END; -- CheckIfPageBreak

  -- Procedure to output var database position
  OutputvSPPosition: PROCEDURE =
  BEGIN
    numString ←
    String.CopyToNewString[Utils.CardToString[tempVarCount],z];
    CheckLength[maxField];
    StatsWrite [numString];
    String.FreeString [z, numString];
    END; -- OutputvSPPosition

  -- Procedure Starts here
  OutputVarTableTitles[];

  -- Process each file until all the database has been output

```

```

WHILE tempVarCount < varCount DO
  fileCount ← fileCount + 1;
  levelCount ← 0;

  -- Move to the beginning of the file
  WHILE tempVarCount < varCount DO
    tempIndex ← vSP[tempVarCount].index;
    IF aSP[tempIndex].idMark = program THEN
      {
        -- Write out current file name
        OutputvSPPosition[];
        numString ←
          String.CopyToString[aSP[tempIndex].inputString,
            g,z];
        CheckLength[maxName];
        StatsWrite [numString];
        String.FreeString [z, numString];
        Stream.PutChar [statsFile, Ascii.CR];
        levelCount ← 0;
        lineCount ← lineCount + 1;
        CheckIfPageBreak[];
        EXIT;
      };
    IF aSP[tempIndex].idMark = monitor THEN
      {
        -- Write out current file name
        OutputvSPPosition[];
        numString ←
          String.CopyToString[aSP[tempIndex].inputString,
            z];
        CheckLength[maxName];
        StatsWrite [numString];
        String.FreeString [z, numString];
        Stream.PutChar [statsFile, Ascii.CR];
        levelCount ← 0;
        lineCount ← lineCount + 1;
        CheckIfPageBreak[];
        EXIT;
      };
    tempVarCount ← tempVarCount + 1;

  ENDOOP; -- tempVarCount

  tempVarCount ← tempVarCount + 1;
  startingIndex ← tempVarCount;
  tempIndex ← vSP[tempVarCount].index;

  --Keep looping until the current file is complete
  WHILE aSP[tempIndex].idMark # program DO
    IF aSP[tempIndex].idMark = monitor THEN EXIT;
    IF tempVarCount > varCount THEN EXIT;

```

```

-- Move forward to the next procedure
WHILE aSP[tempIndex].varMark # levelCount DO
  tempVarCount ← tempVarCount + 1;
  tempIndex ← vSP[tempVarCount].index;
  IF aSP[tempIndex].idMark = program THEN EXIT;
  IF aSP[tempIndex].idMark = monitor THEN EXIT;
  IF tempVarCount > varCount THEN EXIT;
ENDLOOP; -- check procedure

IF aSP[tempIndex].idMark = procedure THEN
  {
    -- Write out current procedure name + location +
    declaration location
    OutputvSPPosition[];
    StatsWrite ["
      numString ←
        String.CopyToString[aSP[tempIndex].inputString,z];
    CheckLength[maxName];
    StatsWrite [numString];
    String.FreeString [z, numString];
    StatsWrite ["
      -- Write out procedure location
      numString ←
        String.CopyToString[Utils.CardToString[tempIndex],
          z];
    CheckLength[maxField];
    StatsWrite [numString];
    String.FreeString [z, numString];
    StatsWrite ["
      -- Write out Declaration Location
      numString ←
        String.CopyToString[Utils.CardToString[aSP[tempIndex]
          ex].declLocation,z];
    CheckLength[maxField];
    StatsWrite [numString];
    String.FreeString [z, numString];
    Stream.PutChar [statsFile, Ascii.CR];
    lineCount ← lineCount + 1;
    CheckIfPageBreak[];
  };

  -- Loop through the variable stack outputting each line
  WHILE aSP[tempIndex].varMark >= levelCount DO

    tempVarCount ← tempVarCount + 1;
    tempIndex ← vSP[tempVarCount].index;
    IF tempVarCount > varCount THEN EXIT;
    IF aSP[tempIndex].idMark = program THEN EXIT;
    IF aSP[tempIndex].idMark = monitor THEN EXIT;
    IF aSP[tempIndex].idMark = variable THEN
      {
        IF aSP[tempIndex].varMark = levelCount THEN

```

```

-- Write out level count
OutputvSPPosition;
StatsWrite[" "];

IF aSP[tempIndex].varMark = 0 THEN
  numString ← String.CopyToNewString["
0",z]
ELSE
  numString ←
    String.CopyToNewString[Utils.CardToSt
ring[aSP[tempIndex].varMark],z];
  CheckLength[maxField];
  StatsWrite [numString];
  String.FreeString [z, numString];
  StatsWrite [" "];

-- Write out variable scope
Stream.PutChar[statsFile,aSP[tempIndex].v
arScope];
StatsWrite [" " "L"];

-- Write out write count
IF vSP[tempVarCount].writeCount = 0 THEN
  numString ← String.CopyToNewString["
0",z]
ELSE
  numString ←
    String.CopyToNewString[Utils.CardToSt
ring[vSP[tempVarCount].writeCount],z];
  CheckLength[maxField];
  StatsWrite [numString];
  String.FreeString [z, numString];
  StatsWrite [" "];

-- Write out read count
IF vSP[tempVarCount].readCount = 0 THEN
  numString ← String.CopyToNewString["
0",z]
ELSE
  numString ←
    String.CopyToNewString[Utils.CardToSt
ring[vSP[tempVarCount].readCount],z];
  CheckLength[maxField];
  StatsWrite [numString];
  String.FreeString [z, numString];
  StatsWrite [" "];

-- Write out Variable Location
index ← vSP[tempVarCount].index;
IF aSP[index].idMark = procedure
  THEN numString ←
    String.CopyToNewString[" " ,z]
  ELSE numString ←
    String.CopyToNewString[Utils.CardTo
String[tempIndex],z];
  CheckLength[maxField];
  StatsWrite [numString];
  String.FreeString [z, numString];

```

```

StatsWrite [" " "L"];

-- Write out Declaration Location
IF aSP[index].idMark = procedure
  THEN numString ←
    String.CopyToNewString[Utils.CardTo
String[tempIndex],z]
  ELSE numString ←
    String.CopyToNewString[Utils.CardTo
String[aSP[tempIndex].declLocation],
z];
IF String.Equal[numString,"0"]
  THEN numString ←
    String.CopyToNewString[" 0",z];
  CheckLength[maxField];
  StatsWrite [numString];
  String.FreeString [z, numString];
  StatsWrite [" " "L"];

-- Write out variable name
numString ←
  String.CopyToNewString[aSP[tempIndex].in
putString,z];
  CheckLength[maxName];
  StatsWrite [numString];
  String.FreeString [z, numString];
  StatsWrite [" " "L"];

-- Write out variable type
SELECT aSP[tempIndex].listMark FROM
callParameter => {
  offset ← 1;
  numString
+String.CopyToNewString[" ,z"];
  WHILE aSP[tempIndex -
offset].symbolNumber # semiColon DO
    IF tempIndex -
offset = 0 THEN EXIT;
    IF aSP[tempIndex
- offset].idMark = procCall THEN
      {
        foundProcCa
11 ← TRUE;
        String [z, numString];
        String.Free
String.CopyToNewString["Call
Param",z];
        EXIT;
      };
      offset ← offset
+ 1;
      ENDDOP; -- aSP
    };
    parameter
=> numString ←
String.CopyToNewString["Formal
Param",z];

```

```

return      => numString ←
String.CopyToNewString["Return
Value",z];
returns    => numString ←
String.CopyToNewString["Formal
Return",z];
ENDCASE    => IF
aSP[tempIndex].idMark = variable
THEN
    numString ←
    String.CopyToNewString["DataBase",
    z]
    ELSE
    numString ←
    String.CopyToNewString["",z];
    CheckLength[maxType];
    StatsWrite [numString];
    String.FreeString [z, numString];
    StatsWrite [" "];
-- Write out procedure call name
IF foundProcCall THEN
    {
        numString ←
        String.CopyToNewString[aSP[tempIndex
        x- offset].inputString,z];
        CheckLength[maxName];
        StatsWrite [numString];
        String.FreeString [z, numString];
        StatsWrite [" "];
    };
    foundProcCall ← FALSE;

```

```

Stream.PutChar [statsFile, Ascii.CR];
lineCount ← lineCount + 1;
CheckIfPageBreak[];

```

```

    };

```

```

ENDLOOP; -- aSP[tempIndex].varMark

```

```

IF levelCount >= fSP[fileCount].procCount THEN EXIT;
tempVarCount ← startingIndex;
tempIndex ← vSP[tempVarCount].index;
levelCount ← levelCount + 1;

```

```

ENDLOOP; --aSP[tempIndex].idMark

```

```

ENDLOOP; -- tempVarCount

```

```

CompletePage[];

```

```

END; -- OutputVariableInfo

```

```

-- OutputVarTableTitles: Puts title at the top of each page
OutputVarTableTitles: PROCEDURE =

```

```

BEGIN
    titleSize: CARDINAL ← 6;

```

```

-- Write identifier titles for export information fields
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["VARIABLE OUPUT TABLE "];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["-----"]

```

```

-----"L];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["vSP Program Name      Procedure Name      Proc
Var   Write Read Var Decl Variable name      Var
Type   Procedure Call Name" ];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["Cnt
Scope Count Count Loca Loc" ];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["-----"]

```

```

-----"L];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["L"];

```

```

lineCount ← lineCount + titleSize

```

```

END; -- OutputVarTableTitles

```

```

-- ConstructContextDiagram: This procedure constructs the diagram
of the
-- highest level of the Data Flow Diagram.
ConstructContextDiagram: PROCEDURE [aIP: LONG POINTER, aSP:
Parser.AtomStackPtr,

```

```

    extSP: LONG POINTER, maxExt:
    CARDINAL,
    fSP: Parser.FileStackPtr,
    MaxFiles: CARDINAL,
    fullStackPtr: Display.FullStackP
    tr, fullCount: CARDINAL] =

```

```

BEGIN
-- Local variables
index: CARDINAL ← 0;
loopCount: CARDINAL ← 0;
tempVarCount: CARDINAL ← 0;
tempDFCount: CARDINAL ← 0;
tempFullCount: CARDINAL ← 0;
found: BOOLEAN ← FALSE;

```

```

Utils.log["
DFCCount ← 0;

loopCount ← loopCount + 1;

-- Look for the external data information
tempVarCount ← 0;

-- The context file is the first impl
DFCCount ← DFCCount + 1;
DFSP[DFCCount].context ← TRUE;
DFSP[DFCCount].nodeName ← 0;
DFSP[DFCCount].nodeType ← nonI;
DFSP[DFCCount].dataName ← vSP[1].index;

-- Find the external variable information
WHILE tempVarCount < varCount DO
    tempVarCount ← tempVarCount + 1;
    index ← vSP[tempVarCount].index;
    IF ASP[index].varScope = 'E' THEN
        {
            tempDFCCount ← DFCCount;
            -- Check to see if we already have that data
            item
            WHILE tempDFCCount > 0 DO
                tempDFCCount ← tempDFCCount - 1;
                IF DFSP[tempDFCCount].dataName =
                    DFSP[DFCCount].dataName THEN
                    { -- Don't add to the database if a
                        repeat
                            found ← TRUE;
                        UNTIL
                            found = FALSE;
                    };
                ENDLOOP; -- tempDFCCount
                IF NOT found THEN
                    {
                        DFCCount ← DFCCount + 1;
                        DFSP[DFCCount].context ← TRUE;
                        DFSP[DFCCount].nodeName ← 0;
                        DFSP[DFCCount].nodeType ← nonI;
                        DFSP[DFCCount].dataName ← index;
                        DFSP[DFCCount].dataName ← External;
                        index ← ASP[loopCount].declLocation;
                        DFSP[DFCCount].connIndex ← index;
                        DFSP[DFCCount].declIn ←
                            FindWhereDecl[ASP, index];
                        DFSP[DFCCount].varCount ← tempVarCount;
                        DFSP[DFCCount].level ← 0;
                        EXIT;
                    };
                found ← FALSE;
            ENDLOOP; -- tempVarCount

            -- Find any SLC information (SLC nodes are input nodes
            at the context level)
        }
    }
end

```

```

WHILE tempFullCount < fullCount DO
    tempFullCount ← tempFullCount + 1;
    IF fullStackPtr[tempFullCount].SLC THEN
        {
            index ←
                fullStackPtr[tempFullCount].declLocation;
            n;
            ASP[index].varScope ← 'S'; -- Entry
            procedure SLC
                DFCCount ← DFCCount + 1;
                DFSP[DFCCount].context ← TRUE;
                DFSP[DFCCount].nodeName ← 0;
                DFSP[DFCCount].nodeType ← nonI;
                DFSP[DFCCount].dataName ← index;
                DFSP[DFCCount].dataName ← External;
                DFSP[DFCCount].level ← 0;
            };

            ENDLOOP; -- tempFullCount

            RollOverDFStack[aIP, ASP, fSP];

            END; -- ConstructContextDiagram

-- ConstructLevelZero: This procedure constructs the diagram of
the
level zero Data Flow Diagram.
ConstructLevelZero: PROCEDURE [aIP: LONG POINTER, ASP:
Parser.AtomStackPtr,
    extSP: LONG POINTER, maxExt:
    CARDINAL,
    fSP: Parser.FileStackPtr,
    MaxFiles: CARDINAL,
    fullStackPtr: Display.FullStackP
    tr, fullCount: CARDINAL] =

BEGIN
    -- Local variables
    index: CARDINAL ← 0;
    level: CARDINAL ← 0;
    tempIndex: CARDINAL ← 0;
    tempCount: CARDINAL ← 0;
    tempVarCount: CARDINAL ← 1;
    listContinue: BOOLEAN ← FALSE;

    Utils.log["    - Constructing level zero diagram.....\n"];
    DFCCount ← 0;

    -- Examine straight line code by
    -- stepping through the Variable index
    -- and picking out items at level zero
    tempIndex ← vSP[tempVarCount].index;
    WHILE tempCount < varCount DO
        listContinue ← FALSE;
    }
end

```

```

tempCount ← tempCount + 1;
IF DFCount # 0 THEN
{
  -- At the start of each new procedure call, go
  -- back to the last one and determine all the
  -- variable connection information
};
IF aSP[tempIndex].listMark = assignment OR
aSP[tempIndex].listMark = 0 THEN
  listContinue ← TRUE;
IF aSP[tempIndex].plevel = 0 AND aSP[tempIndex].idMark =
variable
  AND listContinue THEN
{
  DFCount ← DFCount + 1;
  DFSP[DFCount].level ← 0;
  DFSP[DFCount].SLC ← TRUE;
  DFSP[DFCount].nodeName ← 0; -- Zero signifies Straight
  line code
  DFSP[DFCount].nextHIndex ← vSP[1].index; -- Should be
  first file name
  DFSP[DFCount].nodeType ← Term;
  DFSP[DFCount].dataName ← tempIndex;
  DFSP[DFCount].dataType ← DataBase;
  -- If the variable is part of an array, then the IEatom
  field
  -- contains the root of the array. This is the main
  database
  -- to which this SLC variable is connected
  DetermineConnections[aSP,fullStackPtr];
  DFSP[DFCount].varCount ← tempVarCount;
};
tempVarCount ← tempVarCount + 1;
tempIndex ← vSP[tempVarCount].index;
ENDLOOP; -- aSP

-- Find Level zero nodes, by looking for procedures
-- calls or declarations that are at level 1 and SLC.
-- Procedure calls are stored in the fullStack database
tempCount ← 0;
level ← 1;
WHILE tempCount < fullCount DD
  tempCount ← tempCount + 1;
  -- At the start of each new procedure call, go
  -- back to the last one and determine all the
  -- variable connection information
  IF DFCount # 0 THEN
  {
    RollOverDFStack[aIP,aSP,fSP];
  };
  IF fullStackPtr[tempCount].SLC
  AND NOT fullStackPtr[tempCount].levelRepeat
  THEN
  {
    IF fullStackPtr[tempCount].callLevel = 1 THEN

```

```

{
  DFCount ← DFCount + 1;
  DFSP[DFCount].SLC ← TRUE;
  DFSP[DFCount].level ← 1;
  DFSP[DFCount].nodeName ←
  fullStackPtr[tempCount].callLocation;
  DFSP[DFCount].nextHIndex ← vSP[1].index; -- Should be
  first file name
  IF fullStackPtr[tempCount].callLevel <
  fullStackPtr[tempCount + 1].callLevel
  THEN DFSP[DFCount].nodeType ← nonT
  ELSE DFSP[DFCount].nodeType ← Term;

  -- Go find the parameters
  GoFindCallParams[aIP,aSP,fullCount,fullStackPtr,tempCo
  unt,level,fSP];

  -- Connect to variable stack
  tempVarCount ← fullStackPtr[tempCount].vSPIndex;
  IF tempVarCount # 0 THEN
  {
    tempVarCount ← tempVarCount + 1; -- Move forward
    to first variable
    index ← vSP[tempVarCount].index; -- Get current
    variable location

    -- Loop until all variables for this procedure
    have been parsed
    WHILE aSP[index].idMark = variable DO

      DFCount ← DFCount + 1;
      DFSP[DFCount].level ← 0;
      DFSP[DFCount].nodeName ← 0;
      DFSP[DFCount].dataName ← index;
      SELECT aSP[index].1stMark FROM
        callParameter => DFSP[DFCount].dataType
        + IntCallParam;
        parameter => DFSP[DFCount].dataType
        + FormalParam;
        return => DFSP[DFCount].dataType
        + ReturnValue;
        returns => DFSP[DFCount].dataType
        + FormalReturn;
      ENDCASE
      + DataBase;
      DetermineConnections[aSP,fullStackPtr];
      DFSP[DFCount].varCount ← tempVarCount;
      tempVarCount ← tempVarCount + 1;
      index ← vSP[tempVarCount].index;
    ENDLOOP; -- aSP
  };
};
ENDLOOP; -- tempCount

-- Roll over and output current procedure before leaving

```

```

IF DFCount # 0 THEN
{
  RolloverDFStack[aIP,aSP,fSP];
};

END; -- ConstructLevelZero

-- ConstructLevelDiagrams: This procedure constructs the diagram
of the
-- lower levels of the Data Flow Diagram.
ConstructLevelDiagrams: PROCEDURE [aIP: LONG POINTER, aSP:
  Parser.AtomStackPtr,
    extSP: LONG POINTER, maxExt:
    CARDINAL,
    fSP: Parser.FileStackPtr,
    MaxFiles: CARDINAL,
    fullStackPtr:
    Display.FullStackPtr,
    fullCount: CARDINAL] =

BEGIN
-- Local variables
currentPLevel: CARDINAL + 0;
index: CARDINAL + 0;
level: CARDINAL + 0;
levelCount: CARDINAL + 0;
tempIndex: CARDINAL + 0;
FStempCount: CARDINAL + 0;
tempVarCount: CARDINAL + 1;
procFound: BOOLEAN + TRUE;

Utils.log[" - Constructing lower level diagram.....\n"];
DFCount + 0;

-- Find each level nodes in turn, by looping through for
procedures
-- calls at one level then moving on to the next. Procedure calls
are
-- stored in the fullStack database
WHILE procFound DO
  FStempCount + 0;
  procFound + FALSE;
  levelCount + levelCount + 1;
  WHILE FStempCount < fullCount DO
    FStempCount + FStempCount + 1;
    -- At the start of each new procedure call, go
    -- back to the last one and determine all the
    -- variable connection information
    IF DFCount # 0 THEN
      {
        RolloverDFStack[aIP,aSP,fSP];
      };
    };
  };

```

```

IF fullStackPtr[FStempCount].callLevel = levelCount
AND NOT fullStackPtr[FStempCount].SLC
AND NOT fullStackPtr[FStempCount].levelRepeat
THEN
{
  DFCount + DFCount + 1;
  IF DFCount > maxDFStack THEN
    {
      Utils.log["Error maxDFStack Exceeded ... \n"];
      EXIT;
    };
    procFound + TRUE;
    DFSP[DFCount].level + levelCount;
    DFSP[DFCount].nodeName +
    fullStackPtr[FStempCount].callLocation;
    DFSP[DFCount].nextHIndex +
    FindPreviousLevel[FStempCount, fullStackPtr];
    IF fullStackPtr[FStempCount].callLevel <
    fullStackPtr[FStempCount + 1].callLevel
    THEN DFSP[DFCount].nodeType + nonT
    ELSE DFSP[DFCount].nodeType + Term;
  }

GoFindCallParams[aIP,aSP,fullCount,fullStackPtr,FStemp
Count,levelCount,fSP];

-- Collect variables from the variable stack
tempVarCount + fullStackPtr[FStempCount].vSPIndex; --
Connect to variable stack
IF tempVarCount # 0 THEN
{
  tempVarCount + tempVarCount + 1; -- Move forward
  to first variable
  index + vSP[tempVarCount].index; -- Get current
  variable location
  currentPLevel + aSP[index].Plevel;

  -- Loop until all variable for this procedure have
  been parsed
  WHILE aSP[index].idMark = variable DO
    IF aSP[index].Plevel # currentPlevel THEN
      EXIT;
    }

    -- Check to see if we are at the end of the
    parameter call
    -- list, if so we need to check for a return
    parameter

    DFCount + DFCount + 1;

    -- Continues processing vSP
    DFSP[DFCount].level + levelCount;
    DFSP[DFCount].nodeName + 0;
    DFSP[DFCount].dataName + index;
    SELECT aSP[index].listMark FROM
    callParameter => DFSP[DFCount].dataType
    + IntCallParam;
    parameter => DFSP[DFCount].dataType
    + FormalParam;
  }

```

```

return => DFSP[DFCount].dataType
← ReturnValue;
returns => DFSP[DFCount].dataType
← FormalReturn;
ENDCASE
=> DFSP[DFCount].dataType
← DataBase;
DetermineConnections[aSP,fullStackPtr];
DFSP[DFCount].varCount ← tempVarCount;
tempVarCount ← tempVarCount + 1;
index ← vSP[tempVarCount].index;
IF aSP[index].idMark = procedure THEN
{ -- This is for a procedure declared
  within a procedure
  -- therefore we need to pass over the
  indented one to
  -- get back to the variables of the
  current procedure
  -- that we are examining
  WHILE aSP[index].Plevel >
    currentPlevel DO
    tempVarCount ← tempVarCount + 1;
    index ← vSP[tempVarCount].index;
  ENDOLOOP; -- aSP
};
ENDLOOP; -- aSP
};
ENDLOOP; -- FStempCount
};

-- If the file being processed has all procedures called from
another
-- file then no procedures at level 1 will exist. To prevent all
the
-- higher level procedures being lost procFound is artificially
-- set to TRUE.
IF levelCount = 1 AND procFound = FALSE THEN
  -- Check that there are level 2's around
  {
    FStempCount ← 0;
    WHILE FStempCount < fullCount DO
      FStempCount ← FStempCount + 1;
      IF fullStackPtr[FStempCount].callLevel = 2 THEN
        {
          procFound ← TRUE;
          EXIT;
        };
      ENDLOOP; -- FStempCount
    };
  };
ENDLOOP; --procFound

-- Roll over and output current procedure before leaving
IF DFCount # 0 THEN
{
  RollOverDFStack[aIP,aSP,fSP];
};
};

```

```

END; -- ConstructLevelDiagrams

-- GoFindCallParameters: This procedure adds to the dataflow Stack
-- each of the call parameters used in the procedure call.
GoFindCallParams: PROCEDURE
[aIP:Parser.AtomIndexPtr,aSP:Parser.AtomStackPtr,
fullCount:CARDINAL,
fullStackPtr:Display.FullStackPtr,
tempCount:CARDINAL,level: CARDINAL,
fSP:Parser.FileStackPtr] =

BEGIN
  -- Local Variables
  callIndex: CARDINAL ← 0;
  declIndex: CARDINAL ← 0;
  declParamCount: CARDINAL ← 0;
  paramCount: CARDINAL ← 0;
  tempLevel: CARDINAL ← 0;

  callIndex ← fullStackPtr[tempCount].callLocation;

  -- Check to see if this is a level zero declaration
  IF aSP[callIndex].idMark = procCall THEN
    {
      -- Find the declaration location
      declIndex ← aSP[callIndex].declLocation;
    }
  ELSE
    {
      declIndex ← callIndex;
    };
    IF declIndex # 0 THEN
    {
      -- Move forward to beginning of parameter list
      WHILE aSP[declIndex].listMark # parameter DO
        declIndex ← declIndex + 1;
      IF aSP[declIndex].symbolNumber = semiColon THEN EXIT;
      IF aSP[declIndex].symbolNumber = equals THEN EXIT;
      ENDLOOP; -- aSP

      -- Add the decl locations to the dataflow stack
      WHILE aSP[declIndex].listMark = parameter DO
        declIndex ← declIndex + 1;
      -- Pick out the variables
      IF aSP[declIndex].symbolNumber = semiColon THEN
        EXIT;
      IF aSP[declIndex - 1].symbolNumber = equals THEN
        EXIT;
      IF aSP[declIndex].idMark = variable THEN
        {
          paramCount ← paramCount + 1;
          DFCount ← DFCount + 1;
          tempLevel ← DFSP[DFCount - 1].level;
          DFSP[DFCount].level ← tempLevel;
          DFSP[DFCount].nodeName ← 0;
          DFSP[DFCount].dataType ← formalParam;
        };
      };
    };
  };
END;

```



```

--      DFSP[DFCount].dataName ← declIndex;
--      DFSP[DFCount].level ← level;
--      DFSP[DFCount].parameterCount ← paramCount;
--    };
--  ENDLLOOP; -- asp
--
-- } ELSE
-- {
--   -- Move forward to beginning of call parameter list
--   WHILE asp[callIndex].listMark ≠ callParameter DO
--     callIndex ← callIndex + 1;
--   IF asp[callIndex].symbolNumber = semiColon THEN EXIT;
--   IF asp[callIndex].symbolNumber = equals THEN EXIT;
--   ENDLLOOP; -- asp
--
--   -- Add the call locations to the data flow stack
--   WHILE asp[callIndex].listMark = callParameter DO
--     callIndex ← callIndex + 1;
--     -- Pick out the variables
--     IF asp[callIndex].symbolNumber = semiColon THEN EXIT;
--     IF asp[callIndex - 1].symbolNumber = equals THEN EXIT;
--     IF asp[callIndex].idMark = variable THEN
--       {
--         paramCount ← paramCount + 1;
--         DFCount ← DFCount + 1;
--         tempLevel ← DFSP[DFCount - 1].level;
--         DFSP[DFCount].level ← tempLevel;
--         DFSP[DFCount].nodeName ← 0;
--         DFSP[DFCount].dataType ← CallParam;
--         DFSP[DFCount].dataName ← callIndex;
--         DFSP[DFCount].level ← level;
--         DFSP[DFCount].parameterCount ← paramCount;
--         DetermineConnections[asp,fullStackPtr];
--       };
--     ENDLLOOP; -- asp
--   };
--   END; -- GoFindCallParameters
--
--
-- FindWhereDecl: This procedure looks at the atom stack
-- and determines which procedure or file the variable
-- was declared in.
FindWhereDecl: PROCEDURE [asp: Parser.AtomStackPtr, index: CARDINAL]
  RETURNS[CARDINAL] =
  BEGIN
    -- Local Variables
    loopCount: CARDINAL ← index;
    -- procedure
    WHILE loopCount > 0 DO

```

```

{
  index ← vSP[1].index -- Next higher is the first file
}
ELSE
{
  -- Next higher is the next lower numbered call
  WHILE loopCount > 0 DO
    loopCount ← loopCount - 1;
    currentLevel ← fullStackPtr[loopCount].callLevel;
    IF currentLevel < callLevel THEN EXIT;
  ENDOLOOP; -- loopCount
  index ← fullStackPtr[loopCount].callLocation;
};
RETURN[index];
END; -- FindPreviousLevel

```

```

-- CheckForNonVarData: This procedure looks at known
-- procedure call and determines if there is non
-- variable type data in the call
CheckForNonVarData: PROCEDURE [aSP:Parser.AtomStackPtr,
index:CARDINAL] =
BEGIN

```

```

-- Local Variables

```

```

-- Procedure
IF aSP[index].idMark = procCall
AND aSP[index + 1].idMark = openBracket THEN
{
  WHILE aSP[index].listMark = callParameter DO
    IF aSP[index + 2].symbolNumber = commentString THEN
      {
        numString ← String.CopyToNewString[aSP[index +
        2].inputString,z];
        EXIT;
      }
    ELSE
      {
        index ← index + 1;
      };
    ENDOLOOP; -- aSP
  };
END; -- CheckForNonVarData

```

```

-- DetermineConnections: This procedure looks through the variable
-- information and determines the appropriate connections
information
DetermineConnections: PROCEDURE [aSP:Parser.AtomStackPtr,
fullStackPtr:Display.FullStackPtr] =

```

```

BEGIN
-- Local Variables

```

```

currentIndex: CARDINAL ← 0;
currentDecl: CARDINAL ← 0;
index: CARDINAL ← 0;

-- Procedure Starts here
currentIndex ← DFSP[DFCount].dataName;
currentDecl ← aSP[currentIndex].declLocation;
SELECT DFSP[DFCount].dataType FROM
  FormalParam =>
  {
    -- No further information required
  };
  FormalReturn =>
  {
    -- No further information required
  };
  Database =>
  {
    IF aSP[currentIndex].varScope = 'D' THEN
      {
        DFSP[DFCount].connIndex ←
        currentDecl;
        DFSP[DFCount].declIn ←
        FindWhereDecl[aSP,currentDecl];
      };
    IF aSP[currentIndex].varScope = 'G' THEN
      {
        DFSP[DFCount].connIndex ←
        currentDecl;
        DFSP[DFCount].declIn ←
        FindWhereDecl[aSP,currentDecl];
      };
    IF aSP[currentIndex].varScope = 'L' THEN
      {
        -- No further information needed
      };
    IF aSP[currentIndex].varScope = 'E' THEN
      {
        DFSP[DFCount].connIndex ←
        currentDecl;
        DFSP[DFCount].declIn ←
        FindWhereDecl[aSP,currentDecl];
      };
    IF aSP[currentIndex].varScope = 'M' THEN
      {
        DFSP[DFCount].connIndex ←
        currentIndex;
        IF currentDecl # 0 THEN
          {
            DFSP[DFCount].declIn ← 0;
          };
        };
    IF aSP[currentIndex].varScope = 'N' THEN
      {
        -- No further information needed
      };
    };
    IntCallParam =>
    {
      index ←

```

```

FindCallLocation[aSP,currentIndex];
DFSP[DFCount].connIndex ← index;
index ← aSP[index].declLocation;
DFSP[DFCount].declIn ←
FindWhereDecl[aSP,index];
DFSP[DFCount].alias ← currentDecl;
};
ReturnValue => {
    index ←
FindCallLocation[aSP,currentIndex];
DFSP[DFCount].connIndex ← index;
index ← aSP[index].declLocation;
DFSP[DFCount].declIn ←
FindWhereDecl[aSP,index];
DFSP[DFCount].alias ← currentDecl;
};
ENDCASE; -- DFSP

END; -- DetermineConnections

```

```

-- RollOverDFStack: This procedure swaps out to the output file the
-- current contents of the DFStack, resets the DFStack to zero and
-- then continues on. This is necessary because the DF stack can
-- get so large that a simple fill until done strategy exceeds the
-- capacity of the zone memory.

```

```

RollOverDFStack: PROCEDURE[atp:Parser.AtomIndexPtr, aSP:
Parser.AtomStackPtr, fSP: Parser.FileStackPtr] =

```

```

BEGIN
-- Local variables
loopCount: CARDINAL ← 0;

-- Procedure start
IF DFCount >= maxDFStack THEN DFCount ← maxDFStack;
OutputDFInfo[atp,aSP,fSP];
-- Scrub the old contents of the database
WHILE loopCount < DFCount DO
    loopCount ← loopCount + 1;
    DFSP[loopCount].context ← FALSE;
    DFSP[loopCount].SLC ← FALSE;
    DFSP[loopCount].level ← 0;
    DFSP[loopCount].nodeName ← 0;
    DFSP[loopCount].nodeType ← Dummy;
    DFSP[loopCount].nextHIndex ← 0;
    DFSP[loopCount].dataName ← 0;
    DFSP[loopCount].dataType ← Dummy;
    DFSP[loopCount].connDone ← FALSE;
    DFSP[loopCount].connIndex ← 0;
    DFSP[loopCount].alias ← 0;
    DFSP[loopCount].declIn ← 0;
    DFSP[loopCount].varCount ← 0;

-- Local size information
maxCount: CARDINAL ← 4;
maxName: CARDINAL ← 19;
maxField: CARDINAL ← 5;
maxNodeType: CARDINAL ← 4;
maxType: CARDINAL ← 8;
maxScope: CARDINAL ← 9;

-- Procedure to take care of page endings
CompletePage: PROCEDURE =
    BEGIN
        WHILE lineCount MOD pageSize # 0 DO
            lineCount ← lineCount + 1;
            Stream.PutChar [statsFile, Ascii.CR];
        ENDLOOP; -- Complete Page
        END; -- CompletePage

-- Procedure to check for page breaks
CheckIfPageBreak: PROCEDURE =
    BEGIN
        IF lineCount MOD pageSize = 0 THEN
            OutputOfTitles[];
            END; -- CheckIfPageBreak

-- Procedure Starts here

```

```

ENDLOOP; --loopCount
DFCount ← 0;
END; -- RollOverDFStack

```

```

-- OutputDFInfo: This procedure outputs the Data Flow
Information
-- parsed from the main database, the variable stack and the struct
chart stack
-- previously created. The Output is in sequential order.
OutputDFInfo: PROCEDURE[atp: LONG POINTER, aSP: Parser.AtomStackPtr,
fSP: Parser.FileStackPtr] =

```

```

BEGIN
-- Local variables
firstTime: BOOLEAN ← TRUE;
index: CARDINAL ← 0;
lastProcedure: CARDINAL ← 0;
tempDFCount: CARDINAL ← 0;
tempIndex: CARDINAL ← 0;
tempVarCount: CARDINAL ← 0;
fileCount: CARDINAL ← 1;
procLevel: CARDINAL ← 0;
procLoop: CARDINAL ← 0;
startingIndex: CARDINAL ← 0;
tempNodeName: LONG STRING ← NIL;

```

```

-- Local size information
maxCount: CARDINAL ← 4;
maxName: CARDINAL ← 19;
maxField: CARDINAL ← 5;
maxNodeType: CARDINAL ← 4;
maxType: CARDINAL ← 8;
maxScope: CARDINAL ← 9;

```

```

-- Procedure to take care of page endings
CompletePage: PROCEDURE =
    BEGIN
        WHILE lineCount MOD pageSize # 0 DO
            lineCount ← lineCount + 1;
            Stream.PutChar [statsFile, Ascii.CR];
        ENDLOOP; -- Complete Page
        END; -- CompletePage

```

```

-- Procedure to check for page breaks
CheckIfPageBreak: PROCEDURE =
    BEGIN
        IF lineCount MOD pageSize = 0 THEN
            OutputOfTitles[];
            END; -- CheckIfPageBreak

-- Procedure Starts here

```

```

-- Process all the dataflow database has been output
WHILE tempDFCount < DFCount DO

    tempDFCount ← tempDFCount + 1;

    tempIndex ← DFSP[tempDFCount].nodeName;
    tempVarCount ← DFSP[tempDFCount].varCount;

    IF DFSP[tempDFCount].nodeName # 0 THEN firstTime ← TRUE;

    IF firstTime THEN
    {
        -- Write out data flow level
        IF DFSP[tempDFCount].context THEN
            numString ← String.CopyToNewString[" C",z]
        ELSE
            {
                IF DFSP[tempDFCount].level = 0 THEN
                {
                    numString ← String.CopyToNewString[" 0",z]
                }
                ELSE
                {
                    numString ←
                        String.CopyToNewString[Utils.CardToString[DFSP[
                            tempDFCount].level],z];
                };
            };
            StatsWrite [numString];
            String.FreeString [z, numString];
            StatsWrite [" L"];
        }

        -- Write out node name
        IF DFSP[tempDFCount].context THEN
        {
            numString ← String.CopyToNewString[fSP[1].fileName,z];
        }
        ELSE
        {
            -- Write out current node name (Procedure, file or SLC
            IF DFSP[tempDFCount].SLC AND
            DFSP[tempDFCount].nodeName = 0 THEN
            {
                numString ← String.CopyToNewString["SLC",z];
            }
            ELSE
            {
                index ← DFSP[tempDFCount].nodeName;
                IF index = 0 THEN
                {
                    numString ← String.CopyToNewString["",z];
                }
                ELSE
                {
                    numString ←

```

```

String.CopyToNewString[asp[index].inputString,
g,z];
lastProcedure ← index;
};
};
};
CheckLength[maxName];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" L"];

-- Write out node type
IF DFSP[tempDFCount].nodeType = Term THEN
{
    numString ← String.CopyToNewString["Term",z];
}
ELSE
{
    IF DFSP[tempDFCount].nodeType = nonT THEN
    {
        numString ← String.CopyToNewString["nonT",z];
    }
    ELSE
    {
        numString ← String.CopyToNewString["",z];
    };
};
CheckLength[maxNodeType];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" L"];

firstTime ← FALSE;
}
ELSE
{
    -- Create appropriate indent
    StatsWrite [" L"];
}

-- Write out data name
index ← DFSP[tempDFCount].dataName;
numString ← String.CopyToNewString["",z];
IF index = 0 THEN
{
    -- If no variable with this call check to see if
    other data
    -- is involved, like comments or strings
    IF DFSP[tempDFCount].nodeName # 0 THEN
    {
        index ← DFSP[tempDFCount].nodeName;
        CheckForNonVarData[asp,index];
    };
}
ELSE
{

```

```

numString ←
String.CopyToNewString[asp[index].inputString,z];
};
CheckLength[maxName];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" "L];

-- Write out data type
SELECT DFSP[tempDFCount].dataType FRDM
CallParam => numString ←
String.CopyToNewString["CallParam",z];
Database => numString ←
String.CopyToNewString["Database",z];
DataFlow => numString ←
String.CopyToNewString["DataFlow",z];
External => numString ←
String.CopyToNewString["External",z];
FormalParam => numString ←
String.CopyToNewString["FmlParam",z];
FormalReturn => numString ←
String.CopyToNewString["FmlRetrn",z];
IntCallParam => numString ←
String.CopyToNewString["IntCallP",z];
ReturnValue => numString ←
String.CopyToNewString["RetVal",z];

ENDCASE => numString ← String.CopyToNewString["
",z]; -- DFSP
CheckLength[maxType];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" "L];

-- Write out data scope
index ← DFSP[tempDFCount].dataName;
SELECT asp[index].varScope FRDM
'D => numString ←
String.CopyToNewString["Dereferen",z];
'E => numString ← String.CopyToNewString["External",z];
'G => numString ← String.CopyToNewString["Global",z];
'I => numString ←
String.CopyToNewString["UseFmlCal",z];
'L => numString ← String.CopyToNewString["Local",z];
'M => numString ← String.CopyToNewString["Imported",z];
'N => numString ← String.CopyToNewString["New Zone",z];
'p => numString ← String.CopyToNewString["Public",z];
'R => numString ←
String.CopyToNewString["UseFmlRet",z];
'S => numString ← String.CopyToNewString["Entry",z];
ENDCASE => numString ← String.CopyToNewString["- ",z]; --
asp
CheckLength[maxScope];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" "L];

```

```

-- Write out write count
IF vSP[tempVarCount].writeCount = 0 THEN
numString ← String.CopyToNewString[" 0",z]
ELSE
numString ←
String.CopyToNewString[Utils.CardToString[vSP[tempVarCount]
.writeCount],z];
CheckLength[maxCount];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out read count
IF vSP[tempVarCount].readCount = 0 THEN
numString ← String.CopyToNewString[" 0",z]
ELSE
numString ←
String.CopyToNewString[Utils.CardToString[vSP[tempVarCo
unt].readCount],z];
CheckLength[maxCount];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" "L];

-- Write out connection name
index ← DFSP[tempDFCount].connIndex;
IF index = 0
THEN numString ← String.CopyToNewString["- ",z]
ELSE
{
IF asp[index].varScope = 'M
THEN
{
index ← asp[index].declLocation;
numString ←
String.CopyToNewString[asp[index].input
String,z];
}
ELSE {
numString ←
String.CopyToNewString[asp[index].input
String,z];
};
};
CheckLength[maxName];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" "L];

-- Write out connection declaration location
index ← DFSP[tempDFCount].declIn;
IF index = 0
THEN numString ← String.CopyToNewString["- ",z]
ELSE numString ←

```

```
String.CopyToNewString[aSP[index].inputString,z];
CheckLength[maxName];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" " "L"];

-- Write out connection alias
index ← DFSp[tempDFCount].alias;
IF index = 0
  THEN numString ← String.CopyToNewString["- ",z]
  ELSE numString ←
    String.CopyToNewString[aSP[index].inputString,z];
CheckLength[maxName];
StatsWrite [numString];
String.FreeString [z, numString];

};
firstTime ← FALSE;
Stream.PutChar [statsFile, Ascii.CR];
lineCount ← lineCount + 1;
CheckIfPageBreak[];

ENDLOOP; -- tempDFCount
String.FreeString[z,tempNodeName];

END; -- OutputDFInfo
```

```

Type      Scope      Wr   Rd   Name      Declared in
Alias"L";
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["-----" "L"];
Stream.PutChar [statsFile, Ascii.CR];
lineCount ← lineCount + titleSize
END; -- OutputDFTitles
END. -- DataFlowImpl
```

```
-- Writes the title information at the top of each DataFlow
ChartInfo page
-- Write top of page title
OutputDFTitles: PROCEDURE =
BEGIN
  titleSize: CARDINAL ← 7;

  -- Write identifier titles for export information fields
  Stream.PutChar [statsFile, Ascii.CR];
  StatsWrite ["DATA FLOW DIAGRAM TEXTURAL OUTPUT TABLE "L];
  Stream.PutChar [statsFile, Ascii.CR];
  StatsWrite ["-----" "L"];
  Stream.PutChar [statsFile, Ascii.CR];
  StatsWrite ["      NODE
DATA
CONNECTION"L];
  Stream.PutChar [statsFile, Ascii.CR];
  StatsWrite ["-----" "L"];
  Stream.PutChar [statsFile, Ascii.CR];
  StatsWrite ["Lv1 Node Name      Type      Name
-----" "L];
```

```
-- File: DesignExtractorImpl.mesa - last edit:
-- Lewis:Wbst129:Xerox 1987
-- Bonikowski:HenrB01C:Xerox 28-Apr-89 12:23:11
-- Conley.Henr 10-Mar-89 19:24:12
-- David Egerton:Wbst129:Xerox 13-Nov-93 09:02:51

-- *****
-- File: DesignExtractorImpl.mesa
--
-- Copyright (C) 1989-1992 by Xerox Corporation. All rights reserved.
-- David Egerton:Wbst129:Xerox
--
-- This module sets up the mesa metric design extractor tool, accepts
file name(s)
-- to be processed, then initiates parsing of the contents of the
file. This
-- module is the master controller of operations for design
extractor.
--
-- *****
OIRECTORY

ChangeTable USING [Add, AllEntryNames, CleanUp, Handle, Init,
MakeTable],
Display USING [Control],
Environment USING [wordsPerPage],
Exec USING [AddCommand, ExecProc, LookupCommand, OutputProc,
RemoveCommand],
Format USING [Line, LongNumber, StringProc, Text],
FormSW USING [
AllocateItemDescriptor, BooleanItem, ClientItemsProcType,
CommandItem,
FreeHintsProcType, line0, line1, MenuProcType, nextPlace,
ProcType, sameLine, StringItem],
Heap USING [Create, Delete, systemZone],
HeraldWindow USING [AppendMessage],
MesaTab USING [, -- Parse tables
MFile USING [EnumerateDirectory, EnumerateProc, Error],
MSegment USING [Address, Create, Delete, Handle],
MStream USING [Create, Error, Handle],
parseTable,
Parser USING [ AtomIndex, AtomIndexPtr, AtomProps, AtomStack,
AtomStackPtr, FileStack, FileStackPtr, MaxFiles, MaxString, Parse,
ScanInit],
Process USING [Detach, priorityBackground, SetPriority, Yield],
Put USING [Text],
Runtime USING [BoundsFault, GetTableBase],
Stream USING [Handle, Position, SetPosition],
String USING [AppendStringAndGrow, Copy, CopyToNewString, Empty,
EquivalentSubStrings, FreeString, SubStringDescriptor],
END;
```

```
TextSW USING [ForceOutput],
Token USING [FreeStringHandle, FreeTokenString, Handle, MaybeQuoted,
StringToHandle, UnderminatedQuote],
Tool USING [Create, Destroy, MakeFileSW, MakeFormSW, MakeSwsProc,
UnusedLogName],
ToolDriver USING [Address, NoteSws],
ToolWindow USING [Activate],
UserInput USING [UserAbort],
Utils USING [CardToString, CloseMetFile, ClearMetFile,
MetFileLength, OpenMetFile],
Window USING [Handle];

DesignExtractorImpl: MONITOR
IMPORTS ChangeTable, Display, Exec, Format, FormSW, Heap,
HeraldWindow, MFile,
MesaTab, MSegment, MStream, Parser, Process, Put, Runtime,
Stream,
String, TextSW, Token, Tool, ToolDriver, ToolWindow,
UserInput,
Utils
EXPORTS Utils =
BEGIN
-- Monitor invariant

running: BOOLEAN + FALSE;
created: BOOLEAN + FALSE; -- Ouput test variable

-- External Store Stack
ExternalStackPtr: TYPE = LONG POINTER TO ExternalStack;
ExternalStack: TYPE = ARRAY[0..maxExternal] OF ExternalProps;
ExternalProps: TYPE = RECORD [
extStoreName: LONG STRING + NIL,
extStoreLocation: CARDINAL + 0,
extStoreType: LONG STRING + NIL,
extStoreAction: LONG STRING + NIL];
maxExternal: CARDINAL = 30;
esCount: CARDINAL + 0;

-- DesignExtractor table pointer and indices
fileStackPtr: Parser.FileStackPtr;
fileCount: CARDINAL + 0;
extStackPtr: ExternalStackPtr;

atomStackPtr: Parser.AtomStackPtr;
atomIndexPtr: Parser.AtomIndexPtr;

MaxFiles: CARDINAL = 30;

sh: MSegment.Handle;

zone: PUBLIC UNCOUNTED ZONE + Heap.systemZone;

Enter: ENTRY PROC RETURNS [ok: BOOL] = BEGIN
ENABLE UNWIND => NULL;
RETURN[ok: IF running THEN FALSE ELSE (running + TRUE)];
END; -- Enter
```

```
Exit: ENTRY PROC = BEGIN
  ENABLE UNWIND => NULL;
  running ← FALSE;
END;
```

```
-- Public variables exported to Utils
```

```
permZone, tempZone: PUBLIC UNCOUNTED ZONE ← NIL;
```

```
window, formSW, fileSW: PUBLIC Window.Handle ← NIL;
```

```
-- DesignExtractor command additions
```

```
SCGraph: PUBLIC BOOLEAN ← TRUE;
SCTable: PUBLIC BOOLEAN ← FALSE;
DFDVVariable: PUBLIC BOOLEAN ← FALSE;
DFDTable: PUBLIC BOOLEAN ← TRUE;
rawData: PUBLIC BOOLEAN ← FALSE;
spare: PUBLIC BOOLEAN ← FALSE;
debugg: PUBLIC BOOLEAN ← FALSE;
repeats: PUBLIC BOOLEAN ← FALSE;
```

```
-- Background process control
```

```
backend: PUBLIC BOOLEAN ← FALSE;
encode: PUBLIC BOOLEAN ← FALSE;
background: PUBLIC BOOLEAN ← TRUE;
overlap: PUBLIC CARDINAL ← 10;
```

```
-- Local variables
```

```
totalErrors, totalWarnings: LONG CARDINAL ← 0;
```

```
toolName: LONG STRING ← NIL;
```

```
tablePtr: ParseTable.TableRef ← NIL;
```

```
execCommand: LONG STRING = "DesignExtractor.~"L;
```

```
filename: LONG STRING ← NIL;
```

```
ErrorCode: TYPE = {Other};
```

```
-- Signals
```

```
PublicError: PUBLIC SIGNAL [error: ErrorCode, string: LONG STRING ←
NIL] = CODE;
```

```
-- Procedures
```

```
log: PUBLIC Format.StringProc = BEGIN Put.Text[fileSW, s]; END;
```

```
ExecutiveProc: Exec.ExecProc = BEGIN ToolWindow.Activate>window];
END;

Unload: Exec.ExecProc = BEGIN

  IF ~Enter[] THEN BEGIN
    Format.Text[Exec.OutputProc[h], "DesignExtractor is running.
    Can't unload.\n"~L];
    RETURN[abort];
  END

  ELSE BEGIN
    ENABLE UNWIND => Exit[];

    IF (window # NIL) THEN BEGIN Tool.Destroy>window]; window ← NIL;
    END;

    Exec.RemoveCommand[h, execCommand];

    IF (tempZone # NIL) THEN BEGIN Heap.Delete[tempZone]; tempZone ←
    NIL; END;

    IF (permZone # NIL) THEN BEGIN Heap.Delete[permZone]; permZone ←
    NIL; END;

    IF (atomStackPtr # NIL) THEN BEGIN
      FreeAtomStack[atomStackPtr.sh]; END;

    IF (fileStackPtr # NIL) THEN BEGIN zone.FREE[@fileStackPtr];
    END;

    IF (atomIndexPtr # NIL) THEN BEGIN zone.FREE[@atomIndexPtr];
    END;

    END;

    Exit[];

    END; -- Unload

    CheckForAbort: PUBLIC PROC = BEGIN
      IF UserInput.UserAbort>window] THEN ERROR ABORTED;
    END; -- CheckForAbort

    -- ProcessFile: This procedure initiates the
    -- file parser routines
    ProcessFile: MFile.EnumerateProc =
      BEGIN
        file: MStream.Handle ← NIL;
        errors, warnings: CARDINAL ← 0; -- errors, warnings for this file
        only
        parseZone: UNCOUNTED ZONE ← NIL;
        tempString: LONG STRING;

        CleanUp: PROCEDURE [ok: BOOLEAN + TRUE] =
```



```

BEGIN
    tempASP: CARDINAL ← 0;
    IF file # NIL THEN {
        IF file.delete#NIL THEN file.delete[file];
        file ← NIL;
    }
    log[SELECT TRUE FROM
        ~ok => "... aborted.\n"L,
        errors > 0 => "... errors.\n"L,
        warnings > 0 => "... warnings.\n"L,
        ENDCASE => ".done.\n"L];
    log[" Cumulative Buffer Size = "L];
    log[Utils.CardToString[fileStackPtr[fileCount].endIndex]];
    log[" File Count = "L];
    log[Utils.CardToString[fileCount]];
    log[" ... \n"L];
    log[" LOC this file = "L];
    log[Utils.CardToString[atomStackPtr[atomIndexPtrt].sCount]];
    log[" ... \n"L];
    totalErrors + totalErrors + errors;
    totalWarnings + totalWarnings + warnings;
    END; -- CleanUp

-- Beginning of ProcessFile
BEGIN
    ENABLE
    BEGIN
        MFile.Error, MStream.Error, Runtime.BoundsFault => BEGIN
            log["Unknown file processing error "L];
            GOTO error
        END;
        UNWIND => CleanUp[ok: FALSE];
        END;
    CheckForAbort[];
    file ← MStream.Create[file: fileProc[access: readOnly, release:
        []],
        release: [] ]
        MFile.Error, MStream.Error => {
            log["Cannot open input file "L]; Format.Line[log, name];
            GOTO error};
        log["\nProcessing "L];
        log[name];
        log[" ... \n\L"];
        fileCount + fileCount + 1;
        IF fileCount > Parser.MaxFiles THEN {
            log["\nWarning - DesignExtractor scan aborted, file count
            > ...."L];
            tempString

```

```

+
    String.CopyToNewString[Utils.CardToString[Parser
        ],zone];
    log[tempString];
    String.FreeString[zone, tempString];
    ELSE {
        log["\n\n\L"]
        fileStackPtr[fileCount].fileCount ← fileCount;
        String.FreeString[zone, fileStackPtr[fileCount].fileName];
        fileStackPtr[fileCount].fileName ←
            String.CopyToNewString[name,zone];
        fileStackPtr[fileCount].startIndex ← fileStackPtr[fileCount -
            1].endIndex + 1;
        Stream.SetPosition[sH: file, position: 0];
        parseZone ← Heap.Create[checking: TRUE, ownerChecking: TRUE,
            initial: 15, increment: 15];
        BEGIN
            ENABLE UNWIND =>
                IF parseZone#NIL THEN {Heap.Delete[parseZone]; parseZone ←
                    NIL};
                Parser.ScanInit[tablePtr, file, parseZone];
                [atomStackPtr] ← Parser.Parse[parseZone, atomStackPtr,
                    atomIndexPtr, fileStackPtr, fileCount, rawData];
                fileStackPtr[fileCount].endIndex ← atomIndexPtrt;
            END;
        };
        -- Function ** NOT ** removed below to prevent early
        deallocation
        IF parseZone#NIL THEN {Heap.Delete[parseZone]; parseZone ←
            NIL};
        EXITS
            error => errors + errors + 1;
        END; -- enable
        CleanUp[];
        END; -- ProcessFile
        -- GenTextDisplay: This procedure is the entry
        -- for display processing. At the completion of
        -- this procedure an output file is stored to file.
        -- The files stored to are dependent on the user
        -- selection.
        GenTextDisplay: FormsSW.ProcType = BEGIN
            IF Enter[] THEN BEGIN
                ENABLE UNWIND => Exit[];
                IF tempZone#NIL THEN {
                    Heap.Delete[tempZone]; tempZone ← NIL;
                    -- Make a large zone (works well if we're processing many
                    files).

```

```

tempZone ← Heap.Create[checking: TRUE, ownerChecking: TRUE,
initial:1000, increment: 1000];
[] ← Process.Detach[FORK TextDisplay[]];
END;

END; -- GenTextDisplay

-- TextDisplay: This is the concurrent processing
-- support procedure for GenTextDisplay
TextDisplay: PROCEDURE = BEGIN
ENABLE UNWIND => {
  IF tempZone#NIL THEN {Heap.Delete[tempZone]; tempZone ← NIL};
  Exit[]};
ok: BOOLEAN ← TRUE;

IF background THEN BEGIN
  Process.SetPriority[Process.priorityBackground];
  Process.Yield[]; -- release the notifier
END;

log["\n\n~ ~ ~ Processing Display ~ ~ ~ \n\n"];
BEGIN
ENABLE ABORTED => {ok ← FALSE; CONTINUE};

ok ← CheckSelection[];
IF ok THEN {
  CheckForAbort[];

  [] ← Display.Control[created, atomStackPtr, atomIndexPtr,
extStackPtr, maxExternal,
fileStackPtr, parser.MaxFiles,
rawData, SCGraph, SCTable, DFDVariable, DFDTable,
debugg, repeats];
END; -- enable

IF (tempZone # NIL) THEN BEGIN Heap.Delete[tempZone]; tempZone ←
NIL; END;

log["\n\n~ ~ ~ Display Complete ~ ~ ~ \n\n"];
Exit[];

END; -- TextDisplay

-- CheckSelection: This procedure provides user
-- information about the selection made at the
-- tool user interface.
CheckSelection: PROCEDURE RETURNS[BOOLEAN] =
BEGIN
  ok: BOOLEAN ← TRUE;
  count: CARDINAL ← 0;
  IF rawData THEN

```

```

{
  IF SCGraph OR SCTable OR DFDVariable OR DFDTable THEN
  {
    log["Error - too many display selections set ....\n\n"];
    log["please select according to the following guidelines ..... \n\n"];
    log["If 'Raw Data' required then turn off selections of 'SC Graph', \n\n"];
    log["Variable 'and' DFD Table \n\n"];
    log["OR \n\n"];
    log["If another selection required, turn off 'Raw Data' \n\n"];
    log["Note: Debugg works with any combination. \n\n"];
    ok ← FALSE;
  };
};
IF NOT SCGraph AND NOT SCTable AND NOT DFDVariable AND NOT
DFDTable AND NOT rawData THEN
{
  log["Please select a display output option, none are currently set. \n\n"];
};
RETURN[ok];

END; -- Check Selection

-- Procedure to delete the information in the output files
ClearOutputFile: FormSW.ProcType = BEGIN
  IF Enter[] THEN BEGIN
    ENABLE UNWIND => Exit[];
    [] ← Process.Detach[FORK ClearIt[]];
  END;
END; -- ClearOutputFile

-- Procedure to de allocate databases and re initialize for further
use
ClearIt: PROCEDURE = BEGIN
  ENABLE UNWIND => Exit[];
  ok: BOOLEAN ← TRUE;

  IF background THEN BEGIN
    Process.SetPriority[Process.priorityBackground];
    Process.Yield[]; -- release the notifier
  END;

  log["\n\n~ ~ ~ \n\nCtearing old Extractor data ... \n\n"];
  fileCount ← 0;
  zone.FREE[@fileStackPtr];
  zone.FREE[@filename];

```

```

BEGIN ENABLE
UNWIND => IF tempZone#NIL THEN {Heap.Delete[tempZone]; tempZone
<- NIL};

Utils.OpenMetFile[ 1 MFile.Error, MStream.Error => {
log["Cannot open data file.\n"]; aborted <- TRUE}];

initialFileLength <- Utils.MetFileLength[];

IF NOT aborted AND filename#NIL THEN BEGIN
ENABLE BEGIN
ABORTED => {aborted <- TRUE; CONTINUE};
UNWIND => Utils.CloseMetFile[truncateTo: pos, pos,
initialFileLength];
END;

token: Token.Handle <- Token.StringToHandle[filename];

DO
ENABLE UNWIND => IF token#NIL THEN token <-
token.FreeStringHandle[];

-- Get the next filename in the list.
s: LONG STRING <- token.MaybeQuoted[
data: NIL 1 Token.UnderminatedQuote => RESUME];
IF String.Empty[s] THEN EXIT;

-- Process the file.
BEGIN
ENABLE UNWIND => IF s#NIL THEN s <-
Token.FreeTokenString[s];
[] <- AppendExtensionIfNeeded[to: @s, z: Heap.systemZone]
MFile.EnumerateDirectory[name: s, proc: ProcessFile,
which: filesOnly];
END;

s <- Token.FreeTokenString[s];

ENDLOOP;

token <- token.FreeStringHandle[];

END;

IF aborted THEN
Utils.CloseMetFile[truncateTo: pos, pos: initialFileLength]
ELSE
Utils.CloseMetFile[];

log[IF aborted THEN "\nAnalysis Aborted. "L
ELSE "\nAnalysis Complete. "L];
Format.LongNumber[log, fileCount, []];
log[IF fileCount = 1 THEN " file"L ELSE " files"L];
IF totalErrors > 0 THEN BEGIN

```

```

log["", "L"];
Format.LongNumber[log, totalErrors, []];
log[IF totalErrors = 1 THEN " error" ELSE errors"L"];
END;

IF totalWarnings>0 THEN BEGIN
  log["", "L"];
  Format.LongNumber[log, totalWarnings, []];
  log[IF totalWarnings = 1 THEN " warning" ELSE warnings"L"];
END;

log["\n\nL"];

END; -- enable unwind

Heap.Delete[tempZone];
tempZone ← NIL;
TextSW.ForceOutput[fileSW];

Exit[];

END; -- RunIt

AppendExtensionIfNeeded: PROC [to: LONG POINTER TO LONG STRING,
z: UNCOUNTED ZONE] = BEGIN
-- An extension is needed if the string does not end in one of
-- '.mesa', '#mesa', '*mesa'
minLength: CARDINAL = (".mesa"L).length;

IF to.length>minLength THEN BEGIN
  ext: LONG STRING = "mesa"L;
  toSS: String.SubStringDescriptor ←
    [base: to, offset: to.length-ext.length, length: ext.length];
  extSS: String.SubStringDescriptor ←
    [base: ext, offset: 0, length: ext.length];

  IF String.EquivalentSubStrings[@toSS, @extSS] THEN
    SELECT to[to.length-minLength] FROM
      '...', '#', '*' => RETURN; --no extension needed
    ENDCASE;
  END;

--Extension needed.
String.AppendStringAndGrow[to: to, from: ".mesa"L, z: z];
END; -- AppendExtensionIfNeeded

EnumerateFiles: FormSW.MenuProcType =
BEGIN
  fileList: ChangeTable.Handle ← NIL;
  AddEnumerating: MFile.EnumerateProc = {[] ← fileList.Add[name]};

  -- Init ChangeTable.
  ChangeTable.Init[];
  fileList ← ChangeTable.MakeTable[];

  items[0] ← FormSW.StringItem[
    tag: "Mesa Source Files"L, place: [5, FormSW.line0], string:
      @filename,
    menuProc: EnumerateFiles, inHeap: TRUE];

  items[1] ← FormSW.BooleanItem[
    tag: " SC Graph "L, place: [267, FormSW.sameLine], switch:
      @SCGraph,
    drawBox: TRUE];

  items[2] ← FormSW.BooleanItem[
    tag: " DFD Variable "L, place: [340, FormSW.sameLine], switch:
      @DFDVarTable,
    drawBox: TRUE];

  items[3] ← FormSW.BooleanItem[
    tag: " Raw Data "L, place: [430, FormSW.sameLine], switch:
      @rawData,
    drawBox: TRUE];

  items[4] ← FormSW.CommandItem[
    tag: "Parse"L, place: [5, FormSW.line1], proc: Run];

  items[5] ← FormSW.CommandItem[
    tag: "Display", place: [55, FormSW.sameLine], proc:
      GenTextDisplay];

  items[6] ← FormSW.CommandItem[

```

```

tag: "Clear"L, place: [120, FormSW.sameline], proc:
  ClearOutputFile];

items[7] + FormSW.BooleanItem[
  tag: "SC Table "L, place: [267, FormSW.sameline], switch:
    @SCTable,
  drawBox: TRUE];

items[8] + FormSW.BooleanItem[
  tag: "DFD Table "L, place: [340, FormSW.sameline], switch:
    @DFDTable,
  drawBox: TRUE];

items[9] + FormSW.BooleanItem[
  tag: "Repeats "L, place: [430, FormSW.sameline], switch:
    @Repeats,
  drawBox: TRUE];

RETURN[items: items, freeDesc: TRUE];

END; -- MakeForm

-- This procedure makes the tool window
MakeSWs: Tool.MakeSWsProc = BEGIN
  rootLogName: LONG STRING = "DesignExtractor.log"L;
  logName: LONG STRING = [25];

  addresses: ARRAY [0..2] OF ToolDriver.Address + ALL[[NIL, NIL]];

  formsSW + Tool.MakeFormSW>window: window, formProc: MakeForm];

  Tool.UnusedLogName[unused: logName, root: rootLogName];

  filesSW + Tool.MakeFileSW>window: window, name: logName];

  -- Register the Analyzer with ToolDriver
  addresses + [[name: "formSW", sw: formSW], [name: "fileSW", sw:
    filesSW]];
  ToolDriver.NoteSWs[tool: "DesignExtractor"L, subwindows:
    DESCRIPTOR[addresses]];

  END; -- MakeSWs

Initialize: PROCEDURE =
  BEGIN
    name: LONG STRING + Exec.LookUpCommand[execCommand].name;

    IF (name # NIL) THEN
      BEGIN
        HeraldWindow.AppendMessage["DesignExtractor already loaded"L];
        Heap.systemZone.FREE[aname];
        ERROR ABORTED;
      END;
    END;
  END;

```

```

IF (permZone # NIL) THEN ERROR;

permZone + Heap.Create[checking: TRUE, ownerChecking: TRUE,
  initial: 20, increment: 10];

Exec.AddCommand[name: execCommand, proc: ExecutiveProc, unload:
  Unload];

-- Create the tool name string
BEGIN
  s: LONG STRING + [100];
  String.Copy[to: s, from: "DesignExtractor 1.0 - By: David
    Egerton "L];
  Time.Append[s: s, unpacked: Time.Unpack[Runtime.GetBcdTime[]],
    zone: TRUE];
  String.FreeString[zone, toolName];
  toolName + String.CopyToNewString[s: s, z: permZone];
  END;

window + Tool.Create[
  name: toolName, makeSWsProc: MakeSWs, cmSection:
    "DesignExtractor"L,
  tinyName1: "DE"L, tinyName2: "Analyzer"L];

log[toolName];

END; -- Initialize

-- Procedure to set up atom stack database
AllocateAtomStack: PROCEDURE [elements: CARDINAL]
  RETURNS [sh: MSegment.Handle, asp:
    Parser.AtomStackPtr] =
  BEGIN
    length: LONG POINTER TO CARDINAL;
    words: LONG CARDINAL = LONG[SIZE[Parser.AtomStack]] +
      LONG[elements]*LONG[SIZE[Parser.AtomProps]];
    pages: LONG CARDINAL = (words + LONG[Environment.wordsPerPage]
      - 1)/LONG[Environment.wordsPerPage];
    sh + MSegment.Create[release:[], pages: pages];
    asp + MSegment.Address[sh];
    length + MSegment.Address[sh];
    length + elements;
    END; -- AllocateAtomStack

-- Procedure to de allocate atom stack database
FreeAtomStack: PROCEDURE [asp: Parser.AtomStackPtr, sh:
  MSegment.Handle] =
  BEGIN
    IF asp # MSegment.Address[sh] THEN ERROR; -- Not the right asp
    asp + NIL;
    MSegment.Delete[sh];
    sh + NIL;
  END;

```

```

END;  -- FreeAtomStack

-- ExternalStoreList: This procedure sets up the external input
-- and output location information used to determine the context
-- level of the data flow diagram.
ExternalStoreList: PROCEDURE =
BEGIN
  -- Dummy input, final implementation should be from an external
  -- file that contains these entries. That would make it easier
  -- to
  -- update by the user. However for now the listing are
  -- embedded.
  eSCount ← 1;
  extStackPtr[eSCount].extStoreType ←
    String.CopyToNewString("MStream",zone);
  extStackPtr[eSCount].extStoreAction ←
    String.CopyToNewString("WriteOnly",zone);
END;  -- ExternalStoreList

-- Mainline code
tablePtr ← Runtime.GetTableBase[LOOPHOLE[MesaTab]];
fileStackPtr ← zone.NEW[Parser.FileStack];
atomIndexPtr ← zone.NEW[Parser.AtomIndex];
extStackPtr ← zone.NEW[ExternalStack];
[sh, atomStackPtr] ← AllocateAtomStack[Parser.MaxString];
atomIndexPtr ← 0;
ExternalStoreList[];

Initialize[];

END.  -- PROGRAM DesignExtractor

```

```

-- File: DisplayImpl.mesa - Last Edit:
-- EgertonWbst129:Xerox 13-Nov-1993 09:10:28
--*****
-- File: DisplayImpl.mesa
--
-- Copyright (C) 1989 - 1992 by Xerox Corporation. All rights
reserved.
--
-- David Egerton:Wbst129
--
-- This module controls the selection of the appropriate output
format,
-- Raw Data, Structure Chart, Import Export, Data Flow, Data
Dictionary
-- or Comments
--*****
--*****
DIRECTOR
  Display,
  Parser;

DisplayImpl: PROGRAM
  IMPORTS Display
  EXPORTS Display =

BEGIN
  Control: PUBLIC PROCEDURE [created: BOOLEAN, aStackPtr:
    Parser.AtomStackPtr, aIndexPtr:Parser.AtomIndexPtr, extStackPtr:
    LONG POINTER, maxExternal: CARDINAL, fileStackPtr:
    Parser.FileStackPtr, MaxFiles:CARDINAL, rawData:BOOLEAN,
    SCGraph:BOOLEAN, SCTable:BOOLEAN, DFDVariable:BOOLEAN,
    DFDTable:BOOLEAN, debugg:BOOLEAN, repeats:BOOLEAN] RETURNS
    [BOOLEAN] =

    BEGIN
      result: BOOLEAN ← FALSE;
      IF rawData THEN result ←
        Display.RawData[created,aStackPtr,aIndexPtr,fileStackPtr,
          MaxFiles, debugg];
      IF SCGraph OR SCTable OR DFDVariable OR DFDTable THEN
        result ← Display.StructChart[created, aStackPtr,
          aIndexPtr, extStackPtr,
            maxExternal, fileStackPtr,
            MaxFiles, SCGraph,
            SCTable, DFDVariable,
            DFDTable, debugg, repeats];

      RETURN[result];
    END; -- Control

END. -- Display

```

```

Scan: ActionTag = [FALSE, 0];

inputSymbol: TSymbol;

input: PROC RETURNS [token: Parser.Token];

inputLoc: CAROINAL;
inputValue: Parser.Value ← NIL;

lastToken: Parser.Token;
NullSymbol: TSymbol = 0;

s: Parser.StateStack;
l: Parser.LinkStack;
v: Parser.ValueStack;
top: CAROINAL;

-- ids and idindex added for metrics id collection
ids: PUBLIC Parser.idList ← ALL [NIL];
idindex: PUBLIC CAROINAL ← 1;

q: Parser.ActionStack;
qI: CAROINAL;

tablePtr: ParseTable.TableRef;

-- transition tables for terminal input symbols

tStart: LONG POINTER TO ARRAY State OF TIndex;
tLength: LONG POINTER TO ARRAY State OF CAROINAL;
tSymbol: LONG POINTER TO ARRAY TIndex OF TSymbol;
tAction: LONG POINTER TO ARRAY TIndex OF ActionEntry;

-- transition tables for nonterminal input symbols

nStart: LONG POINTER TO ARRAY NTState OF NTIndex;
nLength: LONG POINTER TO ARRAY NTState OF CAROINAL;
nSymbol: LONG POINTER TO ARRAY NTIndex OF NTSymbol;
nAction: LONG POINTER TO ARRAY NTIndex OF ActionEntry;
ntDefaults: LONG POINTER TO ARRAY NTSymbol OF ActionEntry;

-- production information

prodData: ProdDataRef;

-- For DesignExtractor

aStackPtr: Parser.AtomStackPtr ← NIL;
aIndexPtr: Parser.AtomIndexPtr ← NIL;

ruleValue: CAROINAL ← 0;

-- initialization/termination

```

1


```

z: UNCOUNTED_ZONE ← NIL;

ParseInit: PROC = {
  tablePtr ← Runtime.GetTableBase[LOOPHOLE[MesaTab]];
  -- Parser.ScanInit[tablePtr];
  tStart ← @tablePtr[tablePtr.parseTable.tStart];
  tLength ← @tablePtr[tablePtr.parseTable.tLength];
  tSymbol ← @tablePtr[tablePtr.parseTable.tSymbol];
  tAction ← @tablePtr[tablePtr.parseTable.tAction];
  nStart ← @tablePtr[tablePtr.parseTable.nStart];
  nLength ← @tablePtr[tablePtr.parseTable.nLength];
  nSymbol ← @tablePtr[tablePtr.parseTable.nSymbol];
  nAction ← @tablePtr[tablePtr.parseTable.nAction];
  ntDefaults ← @tablePtr[tablePtr.parseTable.ntDefaults];
  prodData ← @tablePtr[tablePtr.parseTable.prodData];
  s ← NIL; q ← NIL; ExpandStack[512]; ExpandQueue[256];
};

ParseReset: PROC = INLINE {
  EraseQueue[]; EraseStack[]; Parser.ScanReset[];
}

InputLoc: PUBLIC PROC RETURNS [CARDINAL] = {RETURN [inputLoc]};

-- * * * Main Parsing Procedures * * * --

Parse: PUBLIC PROC [cz: UNCOUNTED_ZONE, aStackPtr:
  Parser.AtomStackPtr, aIndexPtr: Parser.AtomIndexPtr,
  fileStackPtr: Parser.FileStackPtr, fileCount: CARDINAL,
  rawData: BOOLEAN] RETURNS [atomSP: LONG POINTER] = {
  currentState: State;
  complete: BOOLEAN;
  nErrors,
  nLines,
  i, valid, m: CARDINAL;
  action: ActionEntry;
  unique: BOOLEAN;
  z ← cz;

  -- stack pointers
  -- True if unique atom

  ParseInit[];
  BEGIN -- of ENABLE
  ENABLE UNWIND => ParseReset[];

  input ← Parser.Atom;

  nErrors ← 0; complete ← TRUE;
  i ← top + valid + 0; qI ← 0;
  s[0] ← currentState + InitialState; lastToken.class ← NullSymbol;
  inputSymbol ← InitialSymbol;
  inputValue ← String.CopyToNewString["L,z];
  inputLoc ← 0;
  ids[1] ← String.CopyToNewString ["L, z];
  idIndex ← 1;

  Utils.log["\nPass1 ... Token identification\n"];

  WHILE currentState ≠ FinalState DO
    BEGIN
      tI: tIndex ← tStart[currentState];
      -- If symbol is id, put on circular stack for metrics
      collection

      IF (inputSymbol = tokenID) THEN
        IF ( NOT Utils.StringEqual[inputValue, ids[idIndex]] ) THEN
          BEGIN
            IF ( idIndex < Parser.maxIds ) THEN
              idIndex ← idIndex + 1;
            ELSE idIndex ← 1;
            ids[idIndex] ← String.CopyToNewString[inputValue, z];
          END;
        FOR tI IN [tI .. tI + tLength[currentState]]
          DO
            SELECT tSymbol[tI] FROM inputSymbol, DefaultMarker => EXIT
          ENDCASE;
          REPEAT
            FINISHED => GO TO SyntaxError;
          ENDLOOP; -- tI

          action ← tAction[tI];
          IF ~action.tag.reduce -- scan or scan reduce entry
          THEN {
            IF qI > 0
              THEN {
                FOR k: CARDINAL IN (valid..i) DO s[k] ← s[top+(k-valid)]
                ENDLOOP;
                [ruleValue] ← Parser.ProcessQueue[qI, top]; qI ← 0;
                [aStackPtr,unique] ←
                  Parser.BaseTable[inputValue,inputSymbol, aIndexPtr,
                    aStackPtr, fileCount, ruleValue];
                IF String.Equal[inputValue,""] THEN EXIT;
                IF aIndexPtr > Parser.MaxString THEN EXIT;
                IF unique THEN aIndexPtr ← aIndexPtr + 1;

                IF (top + valid + i + i+1) >= s.length THEN
                  ExpandStack[256];
                  lastToken.class ← inputSymbol; v[i] ← inputValue; l[i] ←
                    inputLoc;
                  [[inputSymbol, inputValue, inputLoc] + input[]];
                };
              WHILE action.tag ≠ Scan
              DO
                IF qI >= q.length THEN ExpandQueue[256];
                q[qI] ← action; qI ← qI + 1;
                i ← i-action.tag.pLength;
                currentState ← s[If i > valid THEN top+(i-valid) ELSE (valid +
                  i)];
              };
            };
          };
        };
      };
    };
  };

```

```

BEGIN
  lhs: NTSymbol = prodData[action.transition].lhs;
  IF currentState <= LAST[NTState]
  THEN {
    nI: NTIndex + nStart[currentState];
    FOR nI IN [nI..nI+nLength[currentState]]
    DO
      IF lhs = nSymbol[nI] THEN {action ← nAction[nI]; GO TO
        nFound};
      ENOLOOP; -- NTIndex
    action ← ntDefaults[lhs];
  }
  EXITS
    nFound => NULL;
  END;
  i ← i+1;
  ENOLOOP; -- actionTag
  IF (m + top+(i-valid)) >= s.length THEN ExpandStack[256];
  s[m] ← currentState ← action.transition;
  EXITS
    SyntaxError => {
      lastToken.value ← v[top]; lastToken.index ← 1[top];
      top ← top - 1;
      i ← valid + top; qI ← 0; lastToken.class ← NullSymbol;
      currentState ← s[i];
      [[inputSymbol, inputValue, inputLoc] ← input[];
        Utils.log["\n**** Syntax Error at position "L];
        Format.Number[Utils.log, lastToken.index, []];
        Utils.log["\n**** Analyzer aborted.\n\nL];
        EXIT];
      END;
    }
  ENOLOOP; -- CurrentState

  -- [] ← Parser.ProcessQueue[qI, top];
  -- {n: CARDINAL; [nLines, n] ← Parser.ScanStats[]; nErrors ←
  nErrors + n};

  -- DesignExtractor adds for Passes 2 thru 4
  aIndexPtr ← aIndexPtr - 1; -- set to end of file
  Utils.log["\nPass2 ... Context identification
  [] ← Parser.Pass2[inputValue, inputSymbol, aIndexPtr, aStackPtr,
  fileStackPtr, fileCount, ruleValue];

  -- Identify contexts
  Utils.log["\nPass3 ... Resolve identifiers and variables "L];
  [] ← Parser.Pass3[inputValue, inputSymbol, aIndexPtr, aStackPtr,
  fileStackPtr, fileCount, ruleValue];

  -- Count procedure declarations
  Utils.log["\nPass4 ... Count Procedure declarations "L];
  Parser.Pass4[aIndexPtr, aStackPtr, fileStackPtr, fileCount];

  -- Mark the scope of each variable
  Utils.log["\nPass5 ... Determine scope of variables "L];
  Parser.Pass5[aIndexPtr, aStackPtr, fileStackPtr, fileCount];

```

```

END; -- of ENABLE scope
ParseReset[];
RETURN [aStackPtr];

ExpandStack: PROC [delta: NAT] = {
  oldSize: NAT = IF s = NIL THEN 0 ELSE s.length;
  newSize: NAT = oldSize + delta;
  news: Parser.StateStack = z.NEW[Parser.StateSeq[newSize]];
  newL: Parser.LinkStack = z.NEW[Parser.LinkSeq[newSize]];
  newV: Parser.ValueStack = z.NEW[Parser.ValueSeq[newSize]];

  FOR i: NAT IN [0..oldSize) DO
    newS[i] ← s[i]; newL[i] ← 1[i]; newV[i] ← v[i] ENOLOOP;
  EraseStack[];
  s ← news; l ← newL; v ← newV;
  Parser.AssignDescriptors[qd:q, vd:v, ld:l, pp:prodData, z:z];

  EraseStack: PROC = {
    IF s # NIL THEN {z.FREE[0v]; z.FREE[0l]; z.FREE[0s]};

    ExpandQueue: PROC [delta: NAT] = {
      oldSize: NAT = IF q = NIL THEN 0 ELSE q.length;
      newSize: NAT = oldSize + delta;
      newQ: Parser.ActionStack = z.NEW[Parser.ActionSeq[newSize]];
      FOR i: NAT IN [0..oldSize) DO newQ[i] ← q[i] ENOLOOP;
    EraseQueue[];
    q ← newQ;
    Parser.AssignDescriptors[qd:q, vd:v, ld:l, pp:prodData, z:z];

    EraseQueue: PROC = {IF q # NIL THEN z.FREE[0q]};

  }
  END.

```

```
-- File: Pass1Impl.mesa - last edit:
-- Bonikowski:Henr801C:Xerox 27-Apr-89 17:24:47
-- Conley,Henr      6-Mar-89 12:09:00
-- Egerton:Wbst129:Xerox 13-Nov-93 10:17:56
--*****
```

```
-- File: Pass1Impl.mesa
-- David Egerton:Wbst129
-- Copyright (C) 1989 - 1992 by Xerox Corporation. All rights
reserved.
-- This module contains the implementation of the lexical analysis
routines
-- for Pass1 of the DesignExtractor. Originally part of the Mesa
compiler.
```

```
--*****
--*****
```

DIRECTORY

```
Ascii: TYPE USING [ControlZ, CR, FF, NUL, TAB],
Environment: TYPE USING [charsPerWord, charsPerPage, Word,
wordsPerPage],
Format USING [Char, CR, Number, StringProc],
Heap USING [systemZone],
MStream: TYPE USING [EndOf, Handle],
Parser: TYPE USING [AtomIndexPtr, AtomProps, AtomStack,
AtomStackPtr, MaxString, Token, Value],
ParseTable: TYPE USING [HashIndex, HashTableRef, IndexTableRef,
ScanTableRef,
TableRef, TSymbol, VocabularyRef, EndMarker, tokenARROW,
tokenCHAR,
tokenDDT, tokenDOTS, tokenEQUAL, tokenGE, tokenGREATER,
tokenId,
tokenLE, tokenLESS, tokenFLNUM, tokenLNUM, tokenLSTR,
tokenMINUS,
tokenNUM, tokenSTR],
Stream: TYPE USING [GetBlock, GetChar, GetPosition, Handle,
Position,
PutString, SetPosition],
String: TYPE USING [SubStringDescriptor, AppendString,
CopyToNewString, Equal, FreeString, MakeString],
Utils USING [CardToString, CheckForAbort, log];
```

```
Pass1Impl: PROGRAM
```

```
IMPORTS Format, Heap, MStream, Stream, String, Utils
EXPORTS Parser =
```

```
BEGIN
```

```
Pass1Impl.mesa
```

```
13-Nov-93 10:17:45 EST
```

```
OPEN ParseTable;
```

```
hashTab: HashTableRef;
scanTab: ScanTableRef;
vocab: VocabularyRef;
vocabIndex: IndexTableRef;
```

```
stream: Stream.Handle ← NIL; -- the input stream
streamOrigin: Stream.Position;
```

```
TextPages: CARDINAL = 6;
TextWords: CARDINAL = TextPages*Environment.wordsPerPage;
TextChars: CARDINAL = TextWords*Environment.charsPerWord;
TextBuffer: TYPE = PACKED ARRAY [0..TextChars) OF CHARACTER;
```

```
scanZone: UNCOUNTED ZONE ← NIL;
```

```
tb: LONG POINTER TO TextBuffer;
tI, tMax: [0..TextChars];
tDrigin, tLimit: CARDINAL;
tEnded: BOOLEAN;
```

```
FillBuffer: PROCEDURE =
BEGIN
  Utils.CheckForAbort[];
  tOrigin ← tLimit;
  IF tEnded THEN tMax ← 0 ELSE {
    tMax ← stream.GetBlock[LODPHOLE[tB], 0,
TextChars]].bytesTransferred;
    IF tMax < TextChars THEN tEnded ← TRUE;
    tLimit ← tOrigin + tMax;
    IF tMax = 0 THEN {tB[0] ← Ascii.NUL; tMax ← 1};
    tI ← 0;
  }
  END;
```

```
-- buffer: LONG STRING ← NIL; token assembly area
iMax: CARDINAL; -- iMax = buffer.maxlength
desc: String.SubStringDescriptor; -- initial buffer segment
```

```
nLines: CARDINAL; -- line count
nErrors: CARDINAL; -- lexical errors
```

```
hadErrors: BOOLEAN;
BufferOverflow: ERROR = CODE;
```

```
token: Parser.Token;
```

```
-- Additional variables to support the DesignExtractor
atomStackPtr: Parser.AtomStackPtr ← NIL;
```

```
zone: PUBLIC UNCOUNTED ZONE ← Heap.systemZone;
```

```
uniqueAtom: BOOLEAN ← TRUE;
```

```
ExpandBuffer: PROCEDURE =
BEGIN
```

```
-- oldBuffer: LONG STRING ← token.value; «GAB 03/03/89»
```

```

oldBuffer: LONG STRING; «GAB 03/03/89»
token.value.length ← iMax; «GAB 03/03/89»
oldBuffer ← token.value; «GAB 03/03/89»
IF oldBuffer.length > 2000 THEN ERROR BufferOverflow;
token.value ← scanZone.NEW[StringBody[2*oldBuffer.length]];
String.AppendString[token.value, oldBuffer];
-- iMax ← token.value.length + token.value.maxlength;
iMax ← token.value.maxlength;
scanZone.FREE[oldBuffer];
desc.base ← token.value;
ENO;

char: CHARACTER; -- current (most recently scanned) character
qOut: BOOLEAN; -- used to resolved decimal point vs. interval

NextChar: PROC = {
  IF (ti+ti+1) = tMax THEN FillBuffer[]; char ← tB[ti];
  Atom: PUBLIC PROCEDURE [] RETURNS [Parser.Token] =
  BEGIN
    OPEN token;

    preDeclared: BOOLEAN ← FALSE;
    -- value ← ""; «GAB 03/03/89»
    -- value.length ← 0; «GAB 03/03/89»
    initValLength: CAROINAL = 16;
    value ← String.MakeString[z: scanZone, maxLength: initValLength];
    «GAB 03/03/89»
    iMax ← initValLength; «GAB 03/03/89»

    00
    WHILE char IN [Ascii.NUL..' ] 00
      SELECT char FROM
        Ascii.NUL => {
          IF (ti+ti+1) = tMax
            THEN {IF tEnded THEN GO TO EndFile; FillBuffer[]};
          char ← tB[ti];
          IF char = Ascii.NUL THEN GO TO EndFile;
          Ascii.CR => {
            nLines ← nLines + 1;
            IF (ti+ti+1) = tMax
              THEN {IF tEnded THEN GO TO EndFile; FillBuffer[]};
            char ← tB[ti];
            Ascii.FF => {
              IF (ti+ti+1) = tMax
                THEN {IF tEnded THEN GO TO EndFile; FillBuffer[]};
              char ← tB[ti];
              ENOCASE => {
                IF (ti+ti+1) = tMax
                  THEN {IF tEnded THEN GO TO EndFile; FillBuffer[]};
                char ← tB[ti];
                ENOLOOP;
                index ← tOrigin + ti;
                BEGIN

```

```

SELECT char FROM

```

```

'a, 'b, 'c, 'd, 'e, 'f, 'g, 'h, 'i, 'j, 'k, 'l, 'm,
'n, 'o, 'p, 'q, 'r, 's, 't, 'u, 'v, 'w, 'x, 'y, 'z =>

BEGIN
  i: CAROINAL ← 0;
  00
  value[i] ← char;
  value.length ← value.length + 1;
  IF (ti+ti+1) = tMax THEN FillBuffer[];
  char ← tB[ti];
  SELECT char FROM
    IN ['a..'z], IN ['A..'Z], IN ['0..'9'] =>
      IF (i + i+1) >= iMax THEN ExpandBuffer[];
      ENOCASE => EXIT;
      ENOLOOP;
      desc.length ← i+1;
      class ← tokenIO;
      -- IF ~ComData.formatting THEN value.r ←
      SymbolOps.EnterString[@desc];
      GO TO GotNext;
      ENO;

```

```

'A, 'B, 'C, 'O, 'E, 'F, 'G, 'H, 'I, 'J, 'K, 'L, 'M,
'N, 'O, 'P, 'Q, 'R, 'S, 'T, 'U, 'V, 'W, 'X, 'Y, 'Z =>

BEGIN
  i: CAROINAL ← 0;
  uid: BOOLEAN ← TRUE;
  first, last: CAROINAL ← char-0c;
  00
  value[i] ← char;
  value.length ← value.length + 1;
  IF (ti+ti+1) = tMax THEN FillBuffer[];
  char ← tB[ti];
  SELECT char FROM
    IN ['A..'Z'] => {
      last ← char-0c; IF (i + i+1) >= iMax THEN
        ExpandBuffer[];
      IN ['a..'z], IN ['0..'9'] => {
        uid ← FALSE; IF (i + i+1) >= iMax THEN
          ExpandBuffer[];
        ENOCASE => EXIT;
        ENOLOOP;
        i ← i+1;
        desc.length ← i;

```

```

IF uid THEN {
  --hti: Symbols.HTIndex;
  h: HashIndex ← ((first*128-first) + last) MOD
  LAST[HashIndex] + 1;
  j, s1, s2: CAROINAL;
  WHILE (j ← hashTab[h].symbol) # 0 00
    IF vocabIndex[j]-(s2+vocabIndex[j-1]) = i THEN
      FOR s1 IN [0 .. i] 00
        IF value[s1] # vocab.text[s2] THEN EXIT;
        s2 ← s2+1;

```

```

REPEAT
  FINISHED => {class ← j; GO TO GotNext};
ENDLOOP;
IF (h + hashTab[h].link) = 0 THEN EXIT;
ENDLOOP;
-- IF dataPtr.formatting AND
-- (hti ← SymbolOps.FindString[@desc]) # Symbols.HTNull
AND
-- SymbolOps.SearchContext[hti, dataPtr.outerCtx] #
Symbols.ISENull
-- THEN preDeclared ← TRUE;
};
class ← tokenID;
GO TO GotNext;
END;

'0, '1, '2, '3, '4, '5, '6, '7, '8, '9 =>
BEGIN
  valid: BOOLEAN;
  [class, valid] ← CollectNumber[i: 0];
  IF ~valid THEN ScanError[number, index];
  GO TO GotNext;
END;

',' ',' ','#', '+', '*', '/', '.', '0', '1',
'(', ')', '[', ']', '{', '}' =>
BEGIN
  class ← scanTab[char];
  value[0] ← char;
  value.length ← 1;
  GO TO GotNext;
END;

'' =>
BEGIN
  c: CHARACTER;
  valid, advance: BOOLEAN;
  NextChar[];
  [c, valid, advance] ← Escape[];
  IF ~valid THEN ScanError[escape, index + 1];
  class ← tokenCHAR;
  value[0] ← '';
  value[i] ← char;
  value.length ← 2;
  -- IF ~dataPtr.formatting THEN value.r ←
  LiteralOps.Find[c-0c];
  IF advance THEN GO TO GotNext ELSE GO TO GotNext;
END;

'' =>
BEGIN
  i: CARDINAL ← 1;
  valid: BOOLEAN;
  advance: BOOLEAN ← TRUE;

```

```

value[0] ← '';
value.length ← 1;
DO
  IF advance THEN {
    IF (ti+ti+1) = tMax
    THEN {IF tEnded THEN GO TO EOFEnd; FillBuffer[];
      char ← tB[ti]};
    SELECT char FROM
      '' => {
        IF (ti+ti+1) = tMax THEN FillBuffer[];
        char ← tB[ti];
        IF char # '' THEN GO TO QuoteEnd;
        ENDCASE;
        IF i >= iMax THEN ExpandBuffer[
          BufferOverflow => {--ScanError[string, index];--
            i ← 0; CONTINUE}};
        [value[i], valid, advance] ← Escape[];
        i ← i+1;
        -- IF ~valid THEN ScanError[escape, tOrigin + ti];
        REPEAT
          QuoteEnd => NULL; -- {value[i] ← ''; value.length ←
            i+1;
          EOFEnd => {ScanError[string, index]; FillBuffer[]; char
            ← tB[ti]};
        ENDLOOP;
        desc.length ← i;
        -- IF ~dataPtr.formatting THEN value.r ←
        LiteralOps.FindString[@desc];
        IF char = 'l' OR char = 'L' THEN {class ← tokenLSTR; GO TO
          GetNext}
        ELSE {
          class ← tokenSTR;
          IF char = 'g' OR char = 'G' THEN GO TO GetNext
          ELSE GO TO GotNext};
        END;
      }
    }
  }
  char ← Ascii.NUL;
DO
  pChar: CHARACTER = char;
  IF (ti+ti+1) = tMax
  THEN {IF tEnded THEN GO TO EndFile; FillBuffer[];
    char ← tB[ti];
    SELECT char FROM
      '' => {IF pChar = '-' THEN EXIT;
        Ascii.CR => {nLines ← nLines + 1; EXIT};
        ENDCASE;
        ENDLOOP;
        NextChar[];
        END;

```

```

. =>
BEGIN
IF qDot THEN {
qDot ← FALSE; index ← index-1; class ← tokenDOTS;
value ← "...";
-- value[0] ← '.';
-- value[1] ← '.';
value.length ← 2;
GO TO GetNext;
}
NextChar[];
SELECT char FROM
'.' => {class ← tokenDOTS;
value[0] ← '.';
value[1] ← '.';
value.length ← 2;
GO TO GetNext;
}
IN ['0..'9'] => {
valid: BOOLEAN;
value[0] ← '.';
value.length ← 1;
[class, valid] ← CollectNumber[i: 1, float: TRUE];
IF ~valid THEN ScanError[number, index];
GO TO GetNext;
}
ENDCASE => {class ← tokenDOT;
value[0] ← '.';
value.length ← 1;
GO TO GetNext;
}
END;

'.' =>
BEGIN
NextChar[];
IF char = '.' THEN {class ← tokenARROW;
value[0] ← '=';
value[1] ← '.';
value.length ← 2;
GO TO GetNext;
}
ELSE {class ← tokenEQUAL;
value[0] ← '=';
value.length ← 1;
GO TO GetNext;
}
END;

'<' =>
BEGIN
NextChar[];
SELECT char FROM
'.' => {class ← tokenLE;
value[0] ← '<';
value[1] ← '.';
value.length ← 2;
GO TO GetNext;
}
'<' => GO TO ScanComment;
ENDCASE => {class ← tokenLESS;

```

```

value[0] ← '<';
value.length ← 1;
GO TO GotNext;
}
END;

253c => GO TO ScanComment;

'>' =>
BEGIN
NextChar[];
IF char = '=' THEN {class ← tokenGE;
value[0] ← '>';
value[1] ← '=';
value.length ← 2;
GO TO GetNext;
}
ELSE {class ← tokenGREATER;
value[0] ← '>';
value.length ← 1;
GO TO GotNext;
}
END;

-- Character mappings from right set to left
<< This one should be included when ATOM is implemented.
244c => { -- dollar
class ← scanTab['$']; GO TO GetNext;}>>

254c => { -- left arrow
class ← scanTab['_'];
value[0] ← '_';
value.length ← 1;
GO TO GetNext;
}
255c => { -- up arrow
class ← scanTab['^'];
value[0] ← '^';
value.length ← 1;
GO TO GetNext;
}
264c => { -- times
class ← scanTab['*'];
value[0] ← '*';
value.length ← 1;
GO TO GetNext;
}
270c => { -- divide
class ← scanTab['/'];
value[0] ← '/';
value.length ← 1;
GO TO GetNext;
}
ENDCASE =>
BEGIN
class ← scanTab[char];
value[0] ← char;
value.length ← 1;
IF class # 0 THEN GO TO GetNext;
NextChar[];
ScanError[char, index];

```

```

END;

EXITS ScanComment =>

BEGIN
  state: {plain, leftBracket, rightBracket} ← plain;
  nest: CARDINAL ← 1;
DO
  IF (tI+tI+1) = tMax THEN {
    IF tEnded THEN GO TO EndFile; FillBuffer[];
    char ← tB[tI];
    SELECT char FROM
    ' > => SELECT state FROM
      plain, leftBracket => state ← rightBracket;
      rightBracket => {
        state ← plain; nest ← nest - 1; IF nest = 0 THEN
          EXIT;
        ENDCASE;
      }
    ' < => SELECT state FROM
      plain, rightBracket => state ← leftBracket;
      leftBracket => {state ← plain; nest ← nest + 1};
    ENDCASE;
    253c => {state ← plain; nest ← nest + 1};
    273c => {state ← plain; nest ← nest - 1; IF nest = 0 THEN
      EXIT;
    -- Ascii.CR purposely don't count lines here
    ENDCASE => state ← plain;
    ENDOLOOP;
    NextChar[];
  }
END;

END;

REPEAT
  GetNext => {IF (tI+tI+1) = tMax THEN FillBuffer[]; char ←
    tB[tI]};
  GotNext => NULL;
  EndFile => {
    class ← EndMarker; index ← tOrigin + (tI-1);
    value ← ""; value.length ← 0;
    UNTIL tEnded DO FillBuffer[] ENDOLOOP;
    stream
      FillBuffer[]; char ← tB[tI];
    ENDOLOOP;
  }
RETURN [token];

END; -- Atom

-- numerical conversion
LongLit; TYPE = LONG UNSPECIFIED;
endMark; CHARACTER = Ascii.NUL;

CollectNumber: PROC [i: CARDINAL, float: BOOLEAN*FALSE]
  RETURNS [class: TSymbol, valid: BOOLEAN] = {
    hexCount: CARDINAL ← 0;

```

```

tokenFLNUM => NULL;
ENDCASE =>
  IF LOOPHOLE[v, LONG CARDINAL] > maxWord THEN
    NULL
  ELSE {class ← tokenNUM;
        NULL; --token.value ← buffer;--};
RETURN};

Digit: ARRAY CHARACTER ['0..'9] OF CARDINAL = [0,1,2,3,4,5,6,7,8,9];
HexDigit: ARRAY CHARACTER ['A..'F] OF CARDINAL =
[10,11,12,13,14,15];

AppendDecimal: PROC [v: LONG CARDINAL, digit: CHARACTER ['0..'9]]
  RETURNS [newv: LONG CARDINAL, valid: BOOLEAN] = {
  maxv: LONG CARDINAL = 429496729; -- (2**32-1)/10
  maxd: CARDINAL = 5; -- (2**32-1) MOD 10
  d: [0..9] = Digit[digit];
  valid ← v < maxv OR (v = maxv AND d <= maxd);
  newv ← 10*v + d;
  RETURN};

AppendOctal: PROC [v: LONG CARDINAL, digit: CHARACTER ['0..'7]]
  RETURNS [newv: LONG CARDINAL, valid: BOOLEAN] = {
  maxv: LONG CARDINAL = 377777777b; -- (2**32-1)/8
  d: [0..7] = Digit[digit];
  valid ← (v <= maxv);
  newv ← 8*v + d;
  RETURN};

AppendHex: PROC [v: LONG CARDINAL, digit: CHARACTER ['0..'f]]
  RETURNS [newv: LONG CARDINAL, valid: BOOLEAN] = {
  maxv: LONG CARDINAL = 177777777b; -- (2**32-1)/16
  d: [0..15] = IF digit IN ['0..'9] THEN Digit[digit] ELSE
  HexDigit[digit];
  valid ← (v <= maxv);
  newv ← 16*v + d;
  RETURN};

AppendToScale: PROC [v: CARDINAL, digit: CHARACTER ['0..'9]]
  RETURNS [newv: CARDINAL, valid: BOOLEAN] = {
  maxv: CARDINAL = 6553; -- (2**16-1)/10
  maxd: CARDINAL = 5; -- (2**16-1) MOD 10
  d: [0..9] = Digit[digit];
  valid ← v < maxv OR (v = maxv AND d <= maxd);
  newv ← 10*v + d;
  RETURN};

ValidFraction: PROC [v: LONG CARDINAL, digit: CHARACTER ['0..'9]]
  RETURNS [BOOLEAN] = {
  maxv: LONG CARDINAL = 214748364; -- (2**31-1)/10
  maxd: CARDINAL = 7; -- (2**31-1) MOD 10
  RETURN [v < maxv OR (v = maxv AND Digit[digit] <= maxd)];};

ScanDecimal: PROC [s: LONG STRING] RETURNS [value: LongLit, valid:
BOOLEAN+TRUE] = {
  c: CHARACTER;
  i: CARDINAL ← 0;

```

```

  v: LONG CARDINAL ← 0;
  IF s[i] NOT IN ['0..'9] THEN valid ← FALSE;
  WHILE (c ← s[i]) IN ['0..'9] DO
    IF valid THEN [v, valid] ← AppendDecimal[v, c];
    i ← i+1;
  ENDOLOOP;
  IF c = 'd' OR c = 'D' THEN {
    scale: CARDINAL ← 0;
    WHILE (c ← s[i+1]) IN ['0..'9] DO
      IF valid THEN [scale, valid] ← AppendToScale[scale, c];
    ENDOLOOP;
    THROUGH [1..scale] WHILE valid DO
      [v, valid] ← AppendDecimal[v, '0'] ENDOLOOP;
    IF c # endMark THEN valid ← FALSE;
    value ← v;
    RETURN};
  ScanOctal: PROC [s: LONG STRING] RETURNS [value: LongLit, valid:
BOOLEAN+TRUE] = {
  c: CHARACTER;
  i: CARDINAL ← 0;
  v: LONG CARDINAL ← 0;
  IF s[i] NOT IN ['0..'7] THEN valid ← FALSE;
  WHILE (c ← s[i]) IN ['0..'7] DO
    IF valid THEN [v, valid] ← AppendOctal[v, c];
    i ← i+1;
  ENDOLOOP;
  IF c = 'b' OR c = 'B' THEN {
    scale: CARDINAL ← 0;
    WHILE (c ← s[i+1]) IN ['0..'9] DO
      IF valid THEN [scale, valid] ← AppendToScale[scale, c];
    ENDOLOOP;
    THROUGH [1..scale] WHILE valid DO
      [v, valid] ← AppendOctal[v, '0'] ENDOLOOP;
    IF c # endMark THEN valid ← FALSE;
    value ← v;
    RETURN};
  ScanOctalChar: PROC [s: LONG STRING] RETURNS [value: LongLit, valid:
BOOLEAN+TRUE] = {
  c: CHARACTER;
  maxChar: CARDINAL = 377b;
  i: CARDINAL ← 0;
  v: LONG CARDINAL ← 0;
  IF s[i] NOT IN ['0..'7] THEN valid ← FALSE;
  WHILE (c ← s[i]) IN ['0..'7] DO
    IF valid THEN [v, valid] ← AppendOctal[v, c];
    i ← i+1;
  ENDOLOOP;
  IF c = 'c' OR c = 'C' THEN c ← s[i+1] ELSE valid ← FALSE;
  IF c # endMark OR v NOT IN [0..maxChar] THEN valid ← FALSE;
  value ← v;
  RETURN};
  ScanHex: PROC [s: LONG STRING] RETURNS [value: LongLit, valid:
BOOLEAN+TRUE] = {
  c: CHARACTER;
  i: CARDINAL ← 0;

```



```

v: LONG CARDINAL ← 0;
IF s[i] NOT IN ['0..'9] THEN valid ← FALSE;
DO
  SELECT (c ← s[i]) FROM
    IN ['0..'9], IN ['A..'F'] =>
      IF valid THEN [v, valid] ← AppendHex[v, c];
    IN ['a..'f'] =>
      IF valid THEN [v, valid] ← AppendHex[v, c-('a-'A)];
  ENDCASE => EXIT;
  i ← i + 1;
ENDLOOP;
IF c = 'h' OR c = 'H' THEN {
  scale: CARDINAL ← 0;
  WHILE (c ← s[i+1]) IN ['0..'9] DO
    IF valid THEN [scale, valid] ← AppendToScale[scale, c];
  ENDLOOP;
  THROUGH [1..scale] WHILE valid DO
    [v, valid] ← AppendHex[v, '0'] ENDLOOP;
  IF c # endMark THEN valid ← FALSE;
  value ← v;
  RETURN;
}

ScanFloating: PROC [s: LONG STRING] RETURNS [value: LongLit, valid:
BOOLEAN-TRUE] = {
  c: CHARACTER;
  i: CARDINAL ← 0;
  v: LONG CARDINAL ← 0;
  exp: INTEGER ← 0;
  WHILE (c ← s[i]) IN ['0..'9] DO
    valid ← valid AND ValidFraction[v, c];
  IF valid THEN v ← AppendDecimal[v, c].newV
  ELSE exp ← exp + 1;
  i ← i+1;
  ENDLOOP;
  IF c = '.' THEN {
    i ← i+1;
    IF s[i] NOT IN ['0..'9] THEN valid ← FALSE;
    WHILE (c ← s[i]) IN ['0..'9] DO
      valid ← valid AND ValidFraction[v, c];
    IF valid THEN [v, valid] ← AppendDecimal[v, c]; exp ← exp-1}
    ELSE NULL;
    i ← i+1;
    ENDLOOP;
  }
  valid ← TRUE;
  IF c = 'e' OR c = 'E' THEN {
    scale: INTEGER ← 0;
    op: {plus, minus} ← plus;
    i ← i + 1;
    SELECT s[i] FROM
      '+' => i ← i+1;
      '-' => {op ← minus; i ← i+1};
    ENDCASE;
    IF s[i] NOT IN ['0..'9] THEN valid ← FALSE;
    WHILE (c ← s[i]) IN ['0..'9] DO
      IF valid THEN [scale, valid] ← AppendToScale[scale, c];
      i ← i+1;
    ENDLOOP;
    exp ← IF op = plus THEN exp + scale ELSE exp - scale;
  }
}

```

```

need overflow check
IF c # endMark THEN valid ← FALSE;
RETURN;

EnterLit: PROC [v: LongLit, long: BOOLEAN-TRUE] RETURNS
[Parser.Value] = {
  vRep: ARRAY [0..SIZE[LongLit]] OF WORD ← LOOPHOLE[v];
  RETURN [NULL];
};

-- RETURN [[ref[IF long
-- THEN LiteralOps.FindDescriptor[DESCRIPTOR[vRep]]
-- ELSE LiteralOps.Find[vRep[0]]]]];

-- character and string constants
EscapeMark: CHARACTER = 134c;

Escape: PROC RETURNS [c: CHARACTER, valid, advance: BOOLEAN ← TRUE]
=
  BEGIN
    IF char = EscapeMark THEN
      NextChar[];
    IF ( char = '"' ) THEN
      c ← 'x'
    ELSE c ← char;
    RETURN;
  END; -- Escape

-- initialization/finalization
ScanInit: PUBLIC PROC [table: ParseTable.TableRef, infile:
MStream.Handle,
z: UNCOUNTED_ZONE] = {
  scanZone ← z;
  stream ← infile;
  hashTab ← @table[table.scanTable.hashTab];
  scanTab ← @table[table.scanTable.scanTab];
  vocab ← LOOPHOLE[@table[table.scanTable.vocabBody]];
  vocabIndex ← @table[table.scanTable.vocabIndex];
  -- token.value ← scanZone.NEW[StringBody[256]]; «GAB 03/03/89»
  -- iMax ← token.value.length + token.value.maxLength; «GAB
03/03/89»
  desc.base ← token.value; desc.offset ← 0;
  streamOrigin ← stream.GetPosition[];
  tB ← scanZone.NEW[TextBuffer];
  tOrigin ← tLimit ← 0; tMax ← 0; tEnded ← FALSE;
  FillBuffer[]; char ← tB[tI]; qDot ← FALSE;
  nLines ← nErrors ← 0; hadErrors ← FALSE;

  ScanReset: PUBLIC PROC = {tB ← NIL; scanZone ← NIL};
}

```

```

ScanStats: PUBLIC PROC RETURNS [CARDINAL, CARDINAL] = {
  RETURN [nLines, nErrors];
}

```

```
-- error handling
```

```

ResetScanIndex: PUBLIC PROC [index: CARDINAL] RETURNS [success:
  BOOLEAN] = {
  IF ~(index IN [tOrigin .. tLimit]) THEN {
    page: CARDINAL = index/Environment.charsPerPage;
    tOrigin + tLimit < page*Environment.charsPerPage;
    tMax < 0; tEnded < FALSE;
    stream.SetPosition[streamOrigin + tOrigin];
    FillBuffer[];
  }
  tI < index - tOrigin;
  IF tI >= tMax THEN FillBuffer[]; char < tB[tI]; RETURN [TRUE];
}

```

```
ErrorCode: TYPE = {number, string, char, atom, escape};
```

```

ScanError: PROC [code: ErrorCode, tokenIndex: CARDINAL] = {
  nErrors < nErrors + 1;
  ErrorContextPrivate[Utils.log,
    SELECT code FROM
      number => "invalid number"L,
      string => "string unterminated or too long"L,
      char => "invalid character"L,
      atom => "invalid atom"L,
      escape => "invalid escape sequence"L,
      ENDCASE => NIL,
    tokenIndex];
  Format.CR[Utils.log]};

```

```

ErrorContext: PUBLIC PROC [
  to: Stream.Handle, message: LONG STRING, tokenIndex: CARDINAL] = {
  out: Format.StringProc = {Stream.PutString[to,s]};
  ErrorContextPrivate[out: out, message: message, tokenIndex:
    tokenIndex];
};

```

```

ErrorContextPrivate: PUBLIC PROC [
  out: Format.StringProc, message: LONG STRING, tokenIndex:
  CARDINAL] = {
  saveIndex: Stream.Position = stream.GetPosition[];
  origin: Stream.Position = streamOrigin + tokenIndex;
  start, lineIndex: Stream.Position < origin;
  char: CHARACTER;
  n: [1..100];
  FOR n IN [1..100] UNTIL lineIndex = 0 DO
    lineIndex < lineIndex - 1;
    stream.SetPosition[lineIndex];
    IF stream.GetChar[] = Ascii.CR THEN EXIT;
    start < lineIndex;
  ENDOLOOP;
}

```

```

stream.SetPosition[start];
FOR n IN [1..100] UNTIL MStream.EndOf[stream] DO
  char < stream.GetChar[];
  SELECT char FROM
    Ascii.CR, Ascii.ControlZ => EXIT;
    ENDCASE => Format.Char[out, char];
  ENDOLOOP;
  Format.CR[out];
  stream.SetPosition[start];
  UNTIL stream.GetPosition[] = origin OR MStream.EndOf[stream] DO
    char < stream.GetChar[];
    Format.Char[out, IF char = Ascii.TAB THEN Ascii.TAB ELSE
      ];
  ENDOLOOP;
  out["t "L]; out[message]; out[" "L];
  Format.Number[
    out, CARDINAL[origin], [base:10, zeroFill:FALSE, unsigned:TRUE,
      columns:0]];
  Format.Char[out, '']; Format.CR[out];
  stream.SetPosition[saveIndex];
}

```

```

BaseTable: PUBLIC PROCEDURE [inputSymbol: Parser.Value,
  symbolNumber: ParseTable.TSymbol,
  aIP: Parser.AtomIndexPtr,
  aSP: Parser.AtomStackPtr, fileCount: CARDINAL,
  ruleValue: CARDINAL]
  RETURNS[atomSP: Parser.AtomStackPtr, uniqueAtom: BOOLEAN] =

```

```
BEGIN
```

```

-- Set local variables (Also declared in ProgramParserImpl)
boolean: NAT < 202;
cardinal: NAT < 203;
true: NAT < 204;
false: NAT < 205;
nat: NAT < 206;
config: NAT < 207;
string: NAT < 208;

tempString: LONG STRING < NIL;

```

```

IF aIP < 0 THEN -- The first thing to do index is at zero
{
  uniqueAtom < TRUE;
  aSP[aIP] < [0, 0, 0, Program, 0, NIL, '-', 0, NIL,
    0, 0, '-', 0, 0, 0, NIL, NIL, 0, '-', NIL];
}
ELSE
{
  IF aSP[aIP+1].inputString # inputSymbol THEN {
    -- Prime the data base
    aSP[aIP] < [0, 0, 0, Program, 0, NIL, '-', 0, NIL,
      0, 0, '-', 0, 0, 0, NIL, NIL, 0, '-', NIL];
  }
}

```

```

-- Build atom parser data base
ASP[aIPt].BElevel ← 0;
ASP[aIPt].declLocation ← 0;
ASP[aIPt].fileNumber ← fileCount;
ASP[aIPt].fileType ← Program;
ASP[aIPt].idMark ← 0;
ASP[aIPt].IEmark ← '-';
ASP[aIPt].IEatom ←
String.CopyToString["-",zone];
ASP[aIPt].Index ← aIPt;
ASP[aIPt].inputString ←
String.CopyToString[inputSymbol,zone];
ASP[aIPt].listMark ← 0;
ASP[aIPt].Plevel ← 0;
ASP[aIPt].readWrite ← '-';
ASP[aIPt].ruleValue ← ruleValue;
ASP[aIPt].sCount ← 0;
ASP[aIPt].symbolNumber ← symbolNumber;
ASP[aIPt].type ←
String.CopyToString["-",zone];
ASP[aIPt].typeBase ←
String.CopyToString["-",zone];
ASP[aIPt].varMark ← 0;
ASP[aIPt].varScope ← '-';
ASP[aIPt].values ←
String.CopyToString["-",zone];

uniqueAtom ← TRUE;

    Utils.log["lines processed\n\nL"];
    Utils.log["Processing terminated\n\nL"];
};
};
RETURN [aSP,uniqueAtom];
END; -- End of BaseTable
END. -- PROGRAM Pass1Impl

```

```

-- Add in a symbol code designator for Mesa Keywords
-- that have identifier values, ie missing from PGS list.
IF String.Equal[inputSymbol,"BOOLEAN"] THEN
    ASP[aIPt].symbolNumber ← boolean;
IF String.Equal[inputSymbol,"CARDINAL"] THEN
    ASP[aIPt].symbolNumber ← cardinal;
IF String.Equal[inputSymbol,"TRUE"] THEN
    ASP[aIPt].symbolNumber ← true;
IF String.Equal[inputSymbol,"FALSE"] THEN
    ASP[aIPt].symbolNumber ← false;
IF String.Equal[inputSymbol,"NAT"] THEN
    ASP[aIPt].symbolNumber ← nat;
IF String.Equal[inputSymbol,"CONFIGURATION"] THEN
    ASP[aIPt].symbolNumber ← config;
IF String.Equal[inputSymbol,"STRING"] THEN
    ASP[aIPt].symbolNumber ← string;
}
ELSE uniqueAtom ← FALSE; -- Not a new atom
IF uniqueAtom THEN {
    indicator -- Write out progress
    IF aIPt MOD 100 = 0 THEN Utils.log[" L"];
    Utils.log["\n\nWarning - Reached buffer maximum .."];
    tempString ←
String.CopyToString[Utils.CardToString[Parser.MaxS
tring],zone];
    Utils.log[tempString];
    String.FreeString[zone,tempString];
}

```

```

-- Egerton:Wbst129      13-Nov-93  10:19:20
*****
-- File: ProgramParserImpl.mesa
--
-- Oavid Egerton:Wbst129
-- This module uses implements the DesignExtractor parsing routines
  Passes2,3,4.
-- The operation of these procedures is described in the Parser
  definitions
-- module. Basically Pass2 identifies contexts, Pass3 resolves the
  semantics
-- of the non Mesa identifiers and Pass4 counts procedure
  declarations
-- and Pass5 determines the scope of variables.
-- Copyright (C) 1989 - 1993 by Xerox Corporation. All rights
  reserved.
--
-- *****
-- *****
OIRECTORY
Heap,
Parser,
ParseTable,
String,
Utils;

ProgramParserImpl: PROGRAM
  IMPORTS Heap, String, Utils
  EXPORTS Parser =
BEGIN
-- Variable declarations
z: UNCOUNTED ZONE + Heap.systemZone;
uniqueAtom: BOOLEAN + TRUE;

-- Procedure stack count variables, this list is
-- a temporary only, used for finding beginning
-- and ends of procedures
ProcStackPtr: TYPE = LONG POINTER TO ProcedureStack;
ProcedureStack: TYPE = ARRAY[0..maxProc] OF ProcProps;

ProcProps: TYPE = RECORD [
  index: CARDINAL + 0,
  BElevel: CARDINAL + 0];
maxProc: CARDINAL = 300;
procCount: CARDINAL + 0;
procStackPtr: ProcStackPtr + NIL; -- Temporary list of
  procedures, the stack
-- is incremented at a
  declaration and

```

```

-- Procedure List, this list remains active throughout
-- the session. Adding procedures as we go along
ProcListPtr: TYPE = LONG POINTER TO ProcedureList;
ProcedureList: TYPE = ARRAY[0..maxProc] OF ProcProps;
procListCount: CARDINAL + 0;

procListPtr: ProcListPtr + NIL; -- Cumulative list of procedures.
This
-- database is incremented at each
-- procedure.

-- The following variables itemize the numbered Mesa keywords
-- Not all of them just the ones used by the DesignExtractor

atSign: NAT + 28;
array: NAT + 34;
assign: NAT + 14;
assignment: NAT + 209;
begin: NAT + 147;
beginSign: NAT + 146;
boolean: NAT + 202; -- Also declared in PassImpl
callParameter: NAT + 238;
cardinal: NAT + 203; -- Also declared in PassImpl
closeBracket: NAT + 139;
colon: NAT + 11;
comma: NAT + 9;
config: NAT + 207; -- Also declared in PassImpl
definitions: NAT + 45;
directory: NAT + 64;
end: NAT + 141;
entry: NAT + 72;
exports: NAT + 66;
equals: NAT + 15;
false: NAT + 205; -- Also declared in PassImpl
identifier: NAT + 1;
imports: NAT + 65;
machine: NAT + 62;
monitor: NAT + 44;
nat: NAT + 206; -- Also declared in PassImpl
new: NAT + 210;
long: NAT + 48;
openBracket: NAT + 145;
openClause: NAT + 134;
other: NAT + 201;
parameter: NAT + 200;
point: NAT + 27;
pointer: NAT + 32;
program: NAT + 43;
procedure: NAT + 37;
procedureCall: NAT + 237;
public: NAT + 70;
record: NAT + 31;
return: NAT + 117;

```

```

-- decremented at the end

```

```

returns:      NAT ← 57;
semiColon:   NAT ← 10;
sequence:    NAT ← 35;
string:      NAT ← 208; -- Also declared in Pass1Impl
true:        NAT ← 204; -- Also declared in Pass1Impl
type:        NAT ← 49;
typeList:    NAT ← 249;
uncounted:   NAT ← 53;
using:       NAT ← 69;
var:         NAT ← 33;
variable:    NAT ← 500;
variableDecl: NAT ← 501;
zone:        NAT ← 46;

-- Procedure bodies

-- Pass 2 identifies contexts
Pass2: PUBLIC PROCEDURE [inputSymbol: Parser.Value,
symbolNumber: ParseTable.TSymbol, aIP: Parser.AtomIndexPtr, aSP:
Parser.AtomStackPtr, fileStackPtr: Parser.FileStackPtr,
fileCount: CARDINAL, ruleValue: CARDINAL] RETURNS[uniqueAtom: BOOLEAN]
=
BEGIN
-- Local variable declarations
rule: CARDINAL ← 0;
index: CARDINAL ← 0;
tempIndex: CARDINAL ← 0;

-- Procedure begins
index ← fileStackPtr[fileCount - 1].endIndex;
aSP[index].sCount ← 1;
procCount ← 0;
procStackPtr ← z.NEW[ProcedureStack];

WHILE index < aIP DO
have been parsed
  index ← index + 1;
  UpdateFields[aIP, aSP, index];
  rule ← aSP[index].symbolNumber;
  SELECT rule FROM
    10 => BEGIN
      end
      IF procCount > 0 THEN {
        IF aSP[procStackPtr[procCount].index].sCount =
          aSP[index].sCount THEN {
          aSP[index].Plevel ← aSP[index].Plevel -
            1;
          procCount ← procCount - 1;
        }; -- Catch procedure declarations with
          no =
          aSP[index].sCount ← aSP[index].sCount + 1;
        }
      }
    }
  pro

```

```

aSP[index].listMark ← 0;
END;

15 => BEGIN
  -- 15 represents the = symbol
  CheckForUserDefProc[aSP, index];
  IF aSP[index - 1].symbolNumber # type THEN
  { -- Don't increase statement count if the equals
    -- is required for a type declaration
    aSP[index].listMark ← 0; -- For procedure start
    aSP[index].sCount ← aSP[index].sCount + 1;
  };
END;

37 => BEGIN
  tempIndex ← index;
  WHILE aSP[tempIndex].symbolNumber # 1 DO
    tempIndex ← tempIndex - 1;
  ENDLOOP; -- Backup to find procedure identifier
  aSP[tempIndex].idMark ← procedure;
  IF procCount = (maxProc - 1) THEN
  {Utils.log["\nProcedure count overflow
  error.....\n"];
    procCount ← procCount -
      1;
    procListCount ← procListCount + 1; -- Store the
    procedure details in the proc list
    IF procListCount = (maxProc - 1) THEN
    {Utils.log["\nProc list count overflow
    error.....\n"];
      procListCount ← procListCount - 1;
    }
    store Begin/End level
    procListPtr[procListCount].index ← tempIndex; --
    aSP[tempIndex].BElevel ←
    procCount ← procCount + 1; -- Store the procedure
    details in the proc Stack
    procStackPtr[procCount].index ← index; -- store
    Begin/End level
    procStackPtr[procCount].BElevel ← aSP[index].BElevel;
    WHILE tempIndex <= index DO
      aSP[tempIndex].Plevel ← aSP[tempIndex].Plevel
        + 1;
      tempIndex ← tempIndex + 1;
    ENDLOOP; -- Increase procedure by one
    tempIndex ← 0;
  END;

38 => BEGIN
  tempIndex ← index;
  WHILE aSP[tempIndex].symbolNumber # 1 DO
    tempIndex ← tempIndex - 1;
  ENDLOOP; -- Backup to find procedure identifier
  aSP[tempIndex].idMark ← procedure;
  IF procCount = (maxProc - 1) THEN {Utils.log["\nProc
  stack count overflow error.....\n"];
    }
  pro

```

```

        procListCount ← procListCount + 1};
        cCount ← procCount - 1};
        procedure details in the proc list
        IF procListCount = (maxProc - 1) THEN
        {Utils.log["\nProc list count overflow
        error.....\n"];
        proc
        procListPtr[procListCount].index ← tempIndex; --
        store Begin/End level
        procListPtr[procListCount].BElevel ←
        aSP[tempIndex].BElevel;
        procCount ← procCount + 1;
        procStackPtr[procCount].index ← index; -- store
        Begin/End level
        procStackPtr[procCount].BElevel ← aSP[index].BElevel;
        WHILE tempIndex <= index DO
        aSP[tempIndex].Plevel ← aSP[tempIndex].Plevel
        + 1;
        tempIndex ← tempIndex + 1;
        ENDLOOP; -- Increase procedure by one
        tempIndex ← 0;
        END;

64 => BEGIN
        aSP[index].listMark ← directory;
        END;

65 => BEGIN
        aSP[index].listMark ← imports;
        END;

66 => BEGIN
        aSP[index].listMark ← exports;
        END;

122 => BEGIN
        not
        -- 122 represents FREE, this item is
        -- useful as a key word and is
        -- therefore forced
        -- to 1 to make it an identifier
        aSP[index].symbolNumber ← 1;
        END;

139 => BEGIN
        symbol
        -- 139 represents the end of a list, ]
        IF aSP[index].listMark = using THEN
        aSP[index].listMark ← directory
        ELSE aSP[index].listMark ← 0; -- End of non directory
        list
        END;

140 => BEGIN
        -- 140 represents the } symbol
        IF aSP[index].BElevel < 1 THEN ProcessError[];
        aSP[index].BElevel ← aSP[index].BElevel - 1;
        IF procCount # 0 THEN {

```

```

-- Check to see if procedure count decrement
IF procStackPtr[procCount].BElevel =
aSP[index].BElevel
THEN { aSP[index].Plevel ← aSP[index].Plevel -
1;
        procCount ← procCount - 1}; };

        END;

141 => BEGIN
        -- 141 represents the END symbol
        IF aSP[index].BElevel < 1 THEN ProcessError[];
        aSP[index].BElevel ← aSP[index].BElevel - 1;
        IF procCount # 0 THEN {
        -- Check to see if procedure count decrement
        IF procStackPtr[procCount].BElevel =
        aSP[index].BElevel
        THEN { aSP[index].Plevel ← aSP[index].Plevel -
        1;
                procCount ← procCount - 1}; };

        END;

145 => BEGIN
        list [
        -- 145 represents the beginning of a
        -- Determine list type, Parameter, Return or
        other
        37 => aSP[index].listMark ← parameter; --
        PROCEDURE
        38 => aSP[index].listMark ← parameter; --
        PROC
        57 => aSP[index].listMark ← returns; --
        RETURNS
        69 => aSP[index].listMark ← using; --
        USING
        117 => aSP[index].listMark ← return; --
        RETURN
        210 => aSP[index].listMark ← new; --
        NEW
        ENDCASE => aSP[index].listMark ← other; --
        OTHER

        IF aSP[index].listMark = other THEN
        -- Look for an assignment for this list. This is
        when
        -- a procedure call returns and assigns values to
        variables
        -- in a list.
        {
        tempIndex ← index;
        -- Move to the end of the list
        WHILE aSP[tempIndex].symbolNumber # closeBracket
        DO
        IF aSP[tempIndex].symbolNumber = semiColon
        THEN EXIT;
        tempIndex ← tempIndex + 1;
        ENDLOOP; -- aSP

```

```

IF aSP[tempIndex + 1].symbolNumber = assign THEN
{
    aSP[index].listMark + assignment
};
END;

146 => BEGIN
    aSP[index].BElevel + aSP[index].BElevel + 1;
END;

147 => BEGIN
    aSP[index].BElevel + aSP[index].BElevel + 1;
END;

ENDCASE; -- End of rule select
ENDLOOP; -- End of index loop
z.FREE[procStackPtr];
END; -- End of Pass2

ProcessError: PROCEDURE =
BEGIN
    Utils.log["\nError in processing - Pass2....\n\n"];
    Utils.log["Processing Aborted.....\n"];
END; -- ProcessError

UpdateFields: PROCEDURE[aIP:Parser.AtomIndexPtr, aSP:
Parser.AtomStackPtr, index:CARDINAL] =
BEGIN
    aSP[index].BElevel + aSP[index - 1].BElevel;
    aSP[index].idMark + aSP[index - 1].idMark;
    aSP[index].listMark + aSP[index - 1].listMark;
    aSP[index].Plevel + aSP[index - 1].Plevel;
    aSP[index].sCount + aSP[index - 1].sCount;
    aSP[index].varMark + aSP[index - 1].varMark;
    aSP[index].IEmark + '-';
    aSP[index].readWrite + '-';
    aSP[index].varScope + '-';
END; -- UpdateFields

```

```

-- CheckForUserDefProc: In Mesa users can define their own
-- procedure types then import them. This makes life very
-- complicated to determine what is a procedure. This routine
-- attempts to locate and mark them.
CheckForUserDefProc: PROCEDURE [aSP: Parser.AtomStackPtr, index:
CARDINAL] =
BEGIN
    tempIndex: CARDINAL + index;
    offset: CARDINAL + 4;
    publicProc: BOOLEAN + FALSE;

```

```

-- If this equals sign is the end of a list ignore it
IF aSP[index-1].symbolNumber # closeBracket THEN
{
    -- If PUBLIC user declared procedure add to offset
    IF aSP[index - offset].symbolNumber = public THEN
    {
        offset + offset + 1;
        publicProc + TRUE;
    };

    IF aSP[index - offset].symbolNumber = colon THEN
    {
        -- Procedure declaration found
        offset + offset + 1;
        tempIndex + tempIndex - offset;
        aSP[tempIndex].idMark + procedure;
        aSP[tempIndex].IEmark + 'I';
        IF publicProc THEN
        {
            aSP[tempIndex].IEatom +
            String.CopyIoNewString[aSP[tempIndex + 3].inputString,z];
            aSP[tempIndex + 3].symbolNumber + 0; -- Prevent further
            processing
            aSP[tempIndex].type +
            String.CopyIoNewString[aSP[tempIndex + 5].inputString,z];
            aSP[tempIndex + 5].symbolNumber + 0; -- Prevent further
            processing
            aSP[tempIndex].IEmark + 'P';
        }
    }
    ELSE
    {
        aSP[tempIndex].IEatom +
        String.CopyIoNewString[aSP[tempIndex + 2].inputString,z];
        aSP[tempIndex + 2].symbolNumber + 0; -- Prevent further
        processing
        aSP[tempIndex].type +
        String.CopyIoNewString[aSP[tempIndex + 4].inputString,z];
        aSP[tempIndex + 4].symbolNumber + 0; -- Prevent further
        processing
        aSP[tempIndex].IEmark + 'I';
    };
    IF procCount = (maxProc - 1) THEN {Utils.log["\n\nProcedure
count overflow error.....\n"];
        procCount + procCount -
        1};
    procCount + procCount + 1;
    procStackPtr[procCount].index + tempIndex; -- store
    Begin/End level
    procStackPtr[procCount].BElevel + aSP[tempIndex].BElevel;
    procListCount + procListCount + 1;
    procListPtr[procListCount].index + tempIndex; -- store
    Begin/End level
    procListPtr[procListCount].BElevel + aSP[tempIndex].BElevel;
    WHILE tempIndex <= index DO
        aSP[tempIndex].Plevel + aSP[tempIndex].Plevel + 1;
        tempIndex + tempIndex + 1;
    ENDOLOOP; -- Increase procedure by one

```

```

    };
  }; --closeBracket test
END; -- CheckForUserDefProc

-- Pass 3 Resolves identifiers
Pass3: PUBLIC PROCEDURE [inputSymbol: Parser.Value,
symbolNumber: ParseTable.TSymbol,
aIP: Parser.AtomIndexPtr, aSP: Parser.AtomStackPtr,
fileStackPtr: Parser.FileStackPtr, fileCount: CARDINAL,
ruleValue: CARDINAL]
  RETURNS[uniqueAtom: BOOLEAN] =
BEGIN
  -- Local variable declarations
  rule: CARDINAL ← 0;
  index: CARDINAL ← 0;
  tempIndex: CARDINAL ← 0;

  -- Procedure begins
  resolved: BOOLEAN ← FALSE;
  index ← fileStackPtr[fileCount].startIndex;

  -- build a temporary map of procedure identifiers
  procStackPtr ← z.NEW[ProcedureStack];
  tempIndex ← index;
  procCount ← 1;
  WHILE tempIndex < aIP↑ DO
  {
    IF aSP[tempIndex].idMark = procedure THEN
    {
      IF procCount = (maxProc - 1) THEN
        {Utils.log["\nProcedure count overflow
error.....\n"];
        procCount ← procCount - 1;
        procStackPtr[procCount].index ← tempIndex; - store
Begin/End level
        procStackPtr[procCount].BElevel ←
aSP[tempIndex].BElevel;
        procCount ← procCount + 1;
      };
      tempIndex ← tempIndex + 1;
    };
    ENOLOOP;
    tempIndex ← 0;

    WHILE index < aIP↑ DO
      have been parsed
      index ← index + 1;
      resolved ← FALSE;
      rule ← aSP[index].symbolNumber;

      IF rule = 1 THEN {
        IF NOT resolved THEN resolved ←
        {
          FilterNonSpecKeyWords[index,aSP];
          IF NOT resolved THEN resolved ← CheckIfProc0ec1[index,
aSP];
          IF NOT resolved THEN resolved ←
          CheckIfLocalProcCall[index, aSP,
fileStackPtr,fileCount];

          fileStackPtr,fileCount,procCount,procStackPtr];
          IF NOT resolved THEN resolved ← CheckIfVariable0ec1[index,
aSP];
          IF NOT resolved THEN resolved ←
          CheckIfOtherProcCall[index, aSP,
fileStackPtr,fileCount];
          ,procCount,procStackPtr];

          IF NOT resolved THEN resolved ← CheckIfProgram[index,
aSP];
          IF NOT resolved THEN resolved ← CheckListMark[index, aSP,
fileStackPtr,fileCount];
          ,procCount,procStackPtr];

          IF NOT resolved THEN CheckIfVariable[index, aSP];

          IF ProcParameter [aSP,index] THEN {
            resolved ←
            CheckIfVariable0ec1[index, aSP];
            CheckIfVariable[index,
aSP];};
          IF aSP[index - 1].symbolNumber = openClause
          THEN CheckOpenClause[index, aSP,
fileStackPtr,fileCount];
          FindVariablesInLists[index, aSP,fileStackPtr,fileCount];
          ,procCount,procStackPtr];
        };
      };
      ENOLOOP;

      z.FREE[@procStackPtr];

      END; -- Pass3

      FilterNonSpecKeyWords: PROCEDURE [index: CARDINAL,
aSP: Parser.AtomStackPtr] RETURNS [BOOLEAN] =
      BEGIN
        found: BOOLEAN ← FALSE;

        -- Procedure begins here
        IF String.Equal[aSP[index].inputString,"ABORTED"] THEN
          {-- IF symbol is the KEYWORD ABORTED then inhibit
          action
          aSP[index].symbolNumber ← 0;
          found ← TRUE;
          };
        IF String.Equal[aSP[index].inputString,"FREE"] THEN
          {-- IF symbol is the KEYWORD FREE then inhibit action
          aSP[index].symbolNumber ← 0;

```



```

        found ← TRUE;
    };
    IF String.Equal[aSP[index].inputString,"UNWIND"] THEN
    { -- IF symbol is the KEYWORD UNWIND then inhibit
      action
      aSP[index].symbolNumber ← 0;
      found ← TRUE;
    };

    RETURN [ found];

END; -- FilterNonSpecKeyWords

CheckListMark: PROCEDURE [index: CARDINAL, aSP:Parser.AtomStackPtr,
  fileStackPtr:Parser.FileStackPtr,
  fileCount:CARDINAL]
  procCount:CARDINAL, procStackPtr: ProcStackPtr]
  RETURNS [BOOLEAN] =

--
  BEGIN
    found:    BOOLEAN ← TRUE;
    tempIndex: CARDINAL ← index;

    SELECT aSP[index].listMark FROM
      57 => aSP[index].idMark + 57;      -- Return list
      64 => aSP[index].idMark + 64;      -- Directory list
      65 => aSP[index].idMark + 65;      -- Import list
      66 => {aSP[index].idMark + 66;      -- Export list
        fileStackPtr[fileCount].exportName ←
          String.CopyToNewString[aSP[index].inputString,z]};
      117 => aSP[index].idMark + 117;    -- Returns list
      200 => aSP[index].idMark + 200;    -- Parameter list
      201 => {
        IF CheckIfCallList[index, aSP, fileStackPtr,
          fileCount] = FALSE
        THEN aSP[index].idMark + 201; -- Other list
        aSP[index].idMark + 238;      -- Call Parameter
        list
      }

      ENDCASE => found ← FALSE;      -- No list found

    RETURN [ found ];

  END; -- CheckListMark

CheckIfProcDecl: PROCEDURE [index: CARDINAL,
  aSP:Parser.AtomStackPtr] RETURNS [BOOLEAN] =
  BEGIN
    found: BOOLEAN ← FALSE;
    tempIndex: CARDINAL ← index;
    offset: CARDINAL ← 3;

```

```

-- Confirm that this is a procedure declaration
IF aSP[index].idMark = procedure THEN
{
  found ← TRUE;

  -- Correct the procedure indent count and search out
  -- the end of the declaration
  aSP[index].Plevel ← aSP[index - 1].Plevel + 1;

  -- Check if PUBLIC procedure
  IF aSP[index + 2].symbolNumber = public THEN {
    aSP[index].varScope ← 'P;
    offset ← offset + 1};

  -- Check if ENTRY procedure
  IF aSP[index + 2].symbolNumber = entry THEN
    offset ← offset + 1;

  -- Check if this is a one line procedure, if so
  -- mark it and move to end of line.
  -- One line procedures have no begin after the equals
  WHILE aSP[tempIndex].symbolNumber # equals DO
    tempIndex ← tempIndex + 1;
  ENDLOOP; -- tempIndex
  IF aSP[tempIndex + 1].symbolNumber # begin AND
  aSP[tempIndex + 1].symbolNumber # beginSign THEN
  {
    tempIndex ← index;
    WHILE aSP[tempIndex].symbolNumber # semiColon
    DO
      tempIndex ← tempIndex + 1; -- Move to end
      of statement
      aSP[tempIndex].Plevel ← aSP[tempIndex -
        1].Plevel;
    ENDLOOP;
    aSP[tempIndex].Plevel ← aSP[tempIndex].Plevel
    - 1;
  }
  ELSE
  {
    -- Locate the end of the procedure and mark it
    tempIndex ← index;
    WHILE aSP[tempIndex].BElevel = aSP[index +
      1].BElevel DO
      tempIndex ← tempIndex + 1; -- Move forward to
      BEGIN
      aSP[tempIndex].Plevel ← aSP[tempIndex -
        1].Plevel;
      ENDLOOP;
      WHILE aSP[index].BElevel #
      aSP[tempIndex].BElevel DO
        tempIndex ← tempIndex + 1;
        aSP[tempIndex].Plevel ← aSP[tempIndex -
          1].Plevel;
      ENDLOOP; -- Move to end of procedure
      aSP[tempIndex].Plevel ← aSP[tempIndex].Plevel
      - 1;
      aSP[tempIndex + 1].Plevel ←
      aSP[tempIndex].Plevel;
    }
  }
}

```

```

};
};

IF NOT found THEN asp[index].Plevel + asp[index -
1].Plevel;
RETURN [ found];

ENO; -- End CheckIfProcDecl

CheckIfProgram: PROCEDURE [index: CARDINAL,
asp:Parser.AtomStackPtr] RETURNS [BOOLEAN] =
BEGIN
    found: BOOLEAN + TRUE;
    SELECT asp[index + 2].symbolNumber FROM
        43 => {asp[index].idMark + program; -- Program
        header
        asp[1].fileType + Program};
        44 => {asp[index].idMark + monitor; -- Monitor
        header
        asp[1].fileType + Monitor};
        45 => {asp[index].idMark + definitions; -- Definitions
        header
        asp[1].fileType + Definitions};
        207 => { asp[index].idMark + config; --
        Configuration header
        asp[1].fileType + Configuration};

    ENDCASE => found + FALSE; -- No program id found
    RETURN [ found];

ENO; -- CheckIfProgram

CheckIfVariableDecl: PROCEDURE [index: CARDINAL,
asp:Parser.AtomStackPtr]
    RETURNS[BOOLEAN] =
BEGIN
    -- Local variables
    assignFound: BOOLEAN + FALSE;
    found: BOOLEAN + TRUE;
    tempSct: CARDINAL;
    tempIndex: CARDINAL;
    resolved: BOOLEAN + FALSE;

    -- Procedure body begins
    tempIndex + index;
    tempSct + asp[tempIndex].sCount;

    -- If the symbol minus one is a point or closeBracket then
    -- then the variable needs dereferencing to find the root

```

```

-- import interface or variable names. The variable is
-- logged as a side effect of dereferencing
IF asp[index - 1].symbolNumber = point
    THEN Dereference[index, asp];

-- Avoid processing symbols with a point or bracket
-- immediately following the symbol. These items
-- are imported or referenced through array numbers.
IF asp[index + 1].symbolNumber # point THEN
{
    -- Check if type declaration on this source code line
    WHILE tempSct = asp[tempIndex].sCount DO
        -- Check for user declared type fields
        IF asp[index + 1].symbolNumber = colon THEN
        {
            resolved + CheckUserDeclType[index, asp];
            IF resolved THEN EXIT;
        };

        tempIndex + tempIndex + 1;
        IF ListBreak[asp, tempIndex] THEN EXIT;

        SELECT asp[tempIndex].symbolNumber FROM

            31 => {asp[index].idMark + variableDecl;

                asp[index].typeBase +
                String.CopyToNewString[asp[tempIndex].inputSt
                    ring,z];
                resolved + TRUE;
                EXIT};

            32 => {
                resolved + TRUE;
                asp[index].idMark +
                variableDecl;
                IF String.Equal[asp[index].type, "LONG"] THEN
                {
                    asp[index].type +
                    String.CopyToNewString["LONG POINTER",z]
                };
                    -- LONG POINTER
                    IF String.Equal[asp[tempIndex +
                        1].inputString,"To"] THEN
                    {
                        -- LONG POINTER TO
                        IF asp[tempIndex + 3].symbolNumber = point
                        THEN
                        {
                            -- Case where the variable is imported
                            asp[index].IfAtom +
                            String.CopyToNewString[asp[tempIndex +
                                2].inputString,z];

```

```

aSP[tempIndex + 2].symbolNumber ← 0;
aSP[index].values ←
String.CopyToString[aSP[tempIndex +
4].inputString,z];
aSP[tempIndex + 4].symbolNumber ← 0;
}
ELSE -- Not imported
{
IF String.Equal[aSP[tempIndex +
2].inputString,"ARRAY"] THEN
-- ARRAY
{
aSP[index].values ←
String.CopyToString[aSP[tempIndex
+ 2].inputString,z];
IF aSP[tempIndex + 3].symbolNumber =
openBracket THEN
{ -- collect array range data in
brackets
WHILE aSP[tempIndex +
3].symbolNumber # closeBracket 00
IF aSP[tempIndex +
3].symbolNumber = semiColon THEN
EXIT;
String.AppendStringAndGrow[@
aSP[index].values," ",z];
String.AppendStringAndGrow[@
aSP[index].values, aSP[tempIndex +
3].inputString,z];
IF aSP[tempIndex +
3].symbolNumber = variable THEN
aSP[tempIndex
x+ 3].symbolNumber ← 0;
tempIndex ← tempIndex + 1;
ENDLOOP; -- aSP
}
ELSE
{
String.AppendStringAndGrow[@aSP[i
ndex].values," ",z];
String.AppendStringAndGrow[@aSP[i
ndex].values, aSP[tempIndex +
3].inputString,z];
aSP[tempIndex + 3].symbolNumber ←
0;
};
IF String.Equal[aSP[tempIndex +
4].inputString,"OF"] THEN
--
ARRAY XX OF XX
{
String.AppendStringAndGrow[@aS
P[index].values," ",z];
String.AppendStringAndGrow[@aS
P[index].values, aSP[tempIndex +
4].inputString,z];
String.AppendStringAndGrow[@aS
P[index].values," ",z];

```

```

String.AppendStringAndGrow[@aS
P[index].values, aSP[tempIndex +
5].inputString,z];
aSP[tempIndex +
5].symbolNumber ← 0;
};
}
ELSE -- Not imported and not ARRAY
{
aSP[index].values ←
String.CopyToString[aSP[tempIndex
+ 2].inputString,z];
aSP[tempIndex + 2].symbolNumber ← 0;
tempIndex ← tempIndex + 3;
WHILE aSP[tempIndex].symbolNumber #
semiColon 00
IF aSP[tempIndex].symbolNumber =
closeBracket THEN EXIT;
String.AppendStringAndGrow[@aSP[ind
ex].values," ",z];
String.AppendStringAndGrow[@aSP[ind
ex].values,
aSP[tempIndex].inputString,z];
tempIndex ← tempIndex + 1;
ENDLOOP; -- Collect all values info on
line
};
}
ELSE
{
aSP[index].type ←
String.CopyToString[aSP[tempIndex].input
String,z];
aSP[tempIndex].symbolNumber ← 0;
}; -- PTR ONLY
}; -- PTR ONLY

33 => {
resolved ← TRUE;
aSP[index].type ←
String.CopyToString[aSP[tempIndex].inputSt
ring,z]; -- VAR
aSP[index].idMark ← variable0ecl};

34 => {
resolved ← TRUE;
aSP[index].typeBase ←
String.CopyToString[aSP[tempIndex].inputSt
ring,z]; -- ARRAY
aSP[index].idMark ← variable0ecl};

35 => {
resolved ← TRUE;
aSP[index].type ←
String.CopyToString[aSP[tempIndex].inputSt
ring,z]; -- SEQUENCE
aSP[index].idMark ← variable0ecl};

```

```

46 => {
    resolved ← TRUE;
    IF aSP[tempIndex - 1].symbolNumber = 53 THEN
    {
        aSP[index].type ←
        String.CopyToNewString["UNCOUNTED ZONE", z];
        -- UNCOUNTED ZONE
        aSP[index].idMark ← variable0ec1;
    }
    ELSE
    {
        aSP[index].type ←
        String.CopyToNewString[aSP[tempIndex].input
        String, z];
        -- ZONE
        aSP[index].idMark ← variable0ec1;
    };
};

48 => {
    resolved ← TRUE;
    aSP[index].type ←
    String.CopyToNewString[aSP[tempIndex].inputSt
    ring, z];
    -- LONG
    aSP[index].idMark ← variable0ec1;
};

49 => {
    resolved ← TRUE;
    aSP[index].type ←
    String.CopyToNewString[aSP[tempIndex].inputSt
    ring, z];
    -- TYPE
    aSP[index].idMark ← variable0ec1;
    tempIndex ← tempIndex; -- move forward to the
    type information
    IF aSP[tempIndex + 2].symbolNumber = begin
    THEN
    {
        aSP[tempIndex].listMark ← typeList;
        WHILE aSP[tempIndex].symbolNumber # end
        DO
            tempIndex ← tempIndex + 1;
            IF aSP[tempIndex].symbolNumber =
            semiColon THEN EXIT;
            aSP[tempIndex].listMark ← typeList;
        ENDOLOOP;
    };
    IF String.Equal[aSP[tempIndex +
    2].inputString, "ARRAY"] THEN
    ARRAY
    {
        aSP[index].values ←
        String.CopyToNewString[aSP[tempIndex
        + 2].inputString, z];
        IF aSP[tempIndex + 3].symbolNumber =
        openBracket THEN
            { -- collect array range data in

```

```

brackets
    WHILE aSP[tempIndex +
    3].symbolNumber # closeBracket 00
    IF aSP[tempIndex +
    3].symbolNumber = semiColon THEN
    EXIT;
    String.AppendStringAndGrow[@
    aSP[index].values, " ", z];
    String.AppendStringAndGrow[@
    aSP[index].values, aSP[tempIndex +
    3].inputString, z];
    IF aSP[tempIndex +
    3].symbolNumber = variable THEN
        aSP[tempIndex
        x+ 3].symbolNumber ← 0;
        tempIndex ← tempIndex + 1;
    ENDOLOOP; -- aSP
    };
};
IF String.Equal[aSP[tempIndex +
2].inputString, "RECORD"] THEN --
RECORD
{
    aSP[index].type ←
    String.CopyToNewString[aSP[tempIndex
    + 2].inputString, z];
};
};

62 => {
    resolved ← TRUE;
    aSP[index].type ← String.CopyToNewString["MACH
    OEP RECORD", z];
    -- MACHINE
    aSP[index].idMark ← variable0ec1;
};

202 => {
    resolved ← TRUE;
    aSP[index].type ←
    String.CopyToNewString[aSP[tempIndex].inputSt
    ring, z];
    -- BOOLEAN
    aSP[index].idMark ← variable0ec1;
};

203 => {
    resolved ← TRUE;
    aSP[index].type ←
    String.CopyToNewString[aSP[tempIndex].inputSt
    ring, z];
    -- CARDINAL
    aSP[index].idMark ← variable0ec1;
};

206 => {
    resolved ← TRUE;
    IF String.Equal[aSP[tempIndex + 1].type, "OF"]
    THEN {
        -- NAT
        aSP[index].type ← String.CopyToNewString["NAT
        OF", z];
        -- NAT OF
        IF aSP[tempIndex + 3].symbolNumber = point
        THEN {
            -- TO ??

```

```

    ASP[index].IEatom ←
    String.CopyToNewString[ASP[tempIndex +
    2].inputString,z];
    ASP[tempIndex + 2].symbolNumber ← 0;
    ASP[index].values ←
    String.CopyToNewString[ASP[tempIndex +
    4].inputString,z];
    ASP[tempIndex + 4].symbolNumber ← 0;
  ELSE {
    ASP[index].values ←
    String.CopyToNewString[ASP[tempIndex +
    2].inputString,z];
    ASP[tempIndex + 2].symbolNumber ← 0;
  }
  ELSE {
    ASP[index].type ←
    String.CopyToNewString[ASP[tempIndex].in
    putString,z]; -- NAT ONLY
  };
  ASP[index].idMark ← variableDecl;

  208 => {
    resolved ← TRUE;
    ASP[index].type ← String.CopyToNewString["LONG
    STRING",z];
    ASP[index].idMark ← variableDecl;
  }
  ENDCASE;

  ENDLOOP; -- Check for type declarations

  IF resolved THEN
  {
    -- Check if when declared a default value is assigned
    tempIndex ← index;
    WHILE assignFound = FALSE DO
      tempIndex ← tempIndex + 1;
      IF ASP[tempIndex].symbolNumber = closeBracket THEN
        EXIT; -- Abort
      IF ASP[tempIndex].symbolNumber = semiColon THEN EXIT;
      -- Abort
      IF ASP[tempIndex].symbolNumber = equals THEN
        assignFound ← TRUE; -- alternative assign
      IF ASP[tempIndex].symbolNumber = assign THEN
        assignFound ← TRUE; -- found assign
      ENDLOOP; -- Move up to assign character

      IF assignFound = TRUE THEN
      {
        tempIndex ← tempIndex + 1;
        ASP[index].values ←
        String.CopyToNewString[ASP[tempIndex].inputString,z];
        ASP[tempIndex].symbolNumber ← 0; -- Make sure
        identifier not used again later
        tempIndex ← tempIndex + 1;
        WHILE ASP[tempIndex].symbolNumber # semiColon DO
          IF ASP[tempIndex].symbolNumber = closeBracket
          THEN EXIT;

```

```

    String.AppendStringAndGrow[@ASP[index].values,
    ",z"];
    String.AppendStringAndGrow[@ASP[index].values,
    ASP[tempIndex].inputString,z];
    ASP[tempIndex].symbolNumber ← 0; -- Make sure
    identifier not used again later
    tempIndex ← tempIndex + 1;
  ENDLOOP; -- Collect all values info on line
  };

  -- Check to see if there were any other items declared on
  this line. They will
  -- by seen as identifiers with commas immediately after them.

  tempIndex ← index;
  WHILE ASP[tempIndex].symbolNumber = identifier DO
    tempIndex ← tempIndex + 2;
    IF ASP[tempIndex + 1].symbolNumber = comma THEN
    {
      IF ASP[tempIndex].symbolNumber = identifier THEN
      {
        -- Another declaration on the same line
        ASP[tempIndex].idMark ← ASP[index].idMark;
        ASP[tempIndex].varScope ← ASP[index].varScope;
        ASP[tempIndex].readWrite ←
        ASP[index].readWrite;
        ASP[tempIndex].IEmark ← ASP[index].IEmark;
        ASP[tempIndex].IEatom ←
        String.CopyToNewString[ASP[index].IEatom,z];
        ASP[tempIndex].type ←
        String.CopyToNewString[ASP[index].type,z];
        ASP[tempIndex].values ←
        String.CopyToNewString[ASP[index].values,z];
      };
    }
    ENDLOOP; -- ASP
  };
  RETURN[resolved];

  END; -- CheckIfVariableDecl

  -- This procedure looks for variables and marks them and
  -- designates a read or write value
  CheckIfVariable: PROCEDURE [index: CARDINAL,
  ASP:Parser.AtomStackPtr] =
  BEGIN
    -- Local variables
    assignFound: BOOLEAN ← FALSE;
    found: BOOLEAN ← TRUE;
    tempSct: CARDINAL;
    tempIndex: CARDINAL;

```

```
-- Procedure body begins
tempIndex ← index;
tempScnt ← asp[tempIndex].sCount;
```

```
-- Check for Write operation
tempIndex ← index;
tempScnt ← asp[tempIndex].sCount;
IF asp[index + 1].symbolNumber = assign THEN
{
  IF asp[index].idMark # variableDecl THEN
    asp[index].idMark ← variable;
  }
  WRITE OP
  -- STANDARD
}
```

```
WHILE tempScnt = asp[tempIndex].sCount DO -- WHEN TYPE
KNOWN
  tempIndex ← tempIndex + 1;
  IF NOT String.Equal[asp[index].type, "-"] THEN {
    IF asp[tempIndex].symbolNumber = assign THEN {
      IF asp[index].idMark # variableDecl THEN
        asp[index].idMark ← variable;
      }
      ASSIGN
      -- REMOTE
    }
    IF asp[tempIndex - 1].symbolNumber # type THEN
      IF NOT ProcParameter[asp, index] THEN
        IF asp[tempIndex].symbolNumber = equals THEN {
          IF asp[index].idMark # variableDecl THEN
            asp[index].idMark ← variable;
          }
          asp[index].readWrite ← 'W';
        }
        -- TYPE CHECK
      }
    }
  }
ENDLOOP; -- When type known
```

```
-- Check if Read operation
IF asp[index].idMark = 0 THEN {
  IF asp[index].listMark = 0 THEN {
    asp[index].readWrite ← 'R';
    IF asp[index].idMark # variableDecl
      AND asp[index + 1].symbolNumber # point
      THEN asp[index].idMark ← variable;
  }
}
```

```
END; -- CheckIfVariable
```

```
-- FindVariables in Lists: List marks were checked out prior.
However
-- some of these lists have variables of interest. This procedure
marks
-- those variables.
FindVariablesInLists: PROCEDURE[index:CARDINAL,
  asp:Parser.AtomStackPtr,
  fileStackPtr:Parser.FileStackPtr, fileCount:CARDINAL]
=
  procCount: CARDINAL, procStackPtr: ProcStackPtr] =
  BEGIN
```

```
tempProcCount: CARDINAL ← 0;
tempString: LONG STRING ← NIL;
```

```
SELECT asp[index].listMark FROM
```

```
57 => {--RETURNS list (Declaration of variables returned)
  IF asp[index + 1].symbolNumber = colon THEN
    asp[index].idMark ← variable;
  };

117 => {-- RETURN (Actual variable returned)
  asp[index].idMark ← variable;
};

200 => { -- PARAMETER
  IF asp[index + 1].symbolNumber = colon THEN
    asp[index].idMark ← variable;
  };

201 => { -- UNKNOWN LIST
  IF asp[index + 1].symbolNumber # point THEN
  {
    -- First check to see if this could be
    -- a procedure call type. If so we can
    -- call the result a procedure call
    IF asp[index + 1].symbolNumber = colon THEN
    {
      -- Check if there is a procedure match
      WHILE tempProcCount < procListCount DO
        tempProcCount ← tempProcCount + 1;
        tempString ←
          String.CopyToNewString[asp[procListPtr
            r[tempProcCount].index].inputString, z]
        ;
        IF String.Equal[tempString, asp[index +
          2].inputString] THEN
          {
            asp[index+2].idMark ←
              procedureCall;
            asp[index].symbolNumber ← 0;
            String.FreeString[z, tempString];
            EXIT;
          };
          String.FreeString[z, tempString];
        }
      }
      ENDLOOP; -- tempProcCount
    }
    ELSE
    {
      IF asp[index].idMark # procedureCall THEN
        asp[index].idMark ← variable;
      };
    }
  };

209 => { -- PARAMETER FILL LIST OR ASSIGNMENT
  IF asp[index].symbolNumber = identifier THEN
    asp[index].idMark ← variable;
  };
}
```

```

238 => { -- CALLPARAMETER
  IF aSP[index + 1].symbolNumber # point THEN
  {
    -- First check to see if this could be
    -- a procedure call type.If so we can
    -- call the result a procedure call
    IF aSP[index + 1].symbolNumber = colon THEN
    {
      -- Check if there is a procedure match
      WHILE tempProcCount < proclstCount DO
        tempProcCount + tempProcCount + 1;
        tempString +
          String.CopyToNewString[aSP[proclstPtr
            r[tempProcCount].index].inputString,z]
        ;
      IF String.Equal[tempString,aSP[index +
        2].inputString] THEN
      {
        aSP[index+2].idMark +
          procedureCall;
        aSP[index].symbolNumber + 0;
        String.FreeString[z,tempString];
        EXIT;
      };
      String.FreeString[z,tempString];
      ENDOLOOP; -- tempProcCount
    }
    ELSE
    {
      IF aSP[index].idMark # procedureCall THEN
        aSP[index].idMark + variable;
      };
    };
  };
ENDCASE; -- aSP
END; -- FindVariablesInLists

--Check if non local procedure calls
CheckIfDtherProcCall: PROCEDURE[index:CARDINAL,
aSP:Parser.AtomStackPtr,
  fileStackPtr:Parser.FileStackPtr, fileCount:CARDINAL]
procCount: CARDINAL, procStackPtr: ProcStackPtr]
  RETURNS[BOOLEAN] =
BEGIN
tempIndex: CARDINAL + index;
procCall: BOOLEAN + FALSE;
tempProcCount: CARDINAL + 0;

IF aSP[index - 1].symbolNumber # atSign THEN
{
  -- Not a procedure if an at sign
  IF aSP[index + 1].symbolNumber = openBracket THEN {

```

```

  WHILE aSP[tempIndex].symbolNumber # closeBracket DO
    tempIndex + tempIndex + 1;
  ENDOLOOP;

  IF aSP[tempIndex + 1].symbolNumber # point THEN {
    IF aSP[tempIndex + 1].symbolNumber # assign THEN
    {
      -- This is a procedure call unless the variable is declared
      -- in this file
      -- If that is the case the identifier should be marked as
      -- variable. This
      -- part of the routine goes back to the beginning of the
      -- file looking
      -- for a Global variable of this type.
      tempIndex + fileStackPtr[fileCount].startIndex;
      WHILE index > tempIndex DO
        tempIndex + tempIndex + 1;
      IF
        String.Equal[aSP[tempIndex].inputString,aSP[index].in
          putString] THEN
      {
        IF aSP[tempIndex].idMark = variableDecl THEN
        {
          aSP[index].idMark + variable;
          aSP[index].varScope + 'G;
          EXIT;
        };
      };
      ENDOLOOP; -- index
      -- IF tempIndex and index are the same then the variable
      -- was not found
      -- and this is clearly a procedure.
      IF tempIndex = index THEN
      {
        procCall + TRUE;
        aSP[index].idMark + procedureCall;
        MarkCallList[index, aSP];
      }
      IF aSP[index - 1].symbolNumber = point THEN
      {
        aSP[index].IEatom +
          String.CopyToNewString[aSP[index -
            2].inputString,z];
        aSP[index - 2].symbolNumber + 0;
      };
    };
  };
};
};
RETURN[procCall];
END; -- CheckIfDtherProcCall

--Check if local procedure call
CheckIfLocalProcCall: PROCEDURE[index:CARDINAL,
aSP:Parser.AtomStackPtr,
```

```

fileStackPtr.Parser.FileStackPtr, fileCount:CARDINAL]
procCount: CARDINAL, procStackPtr: ProcStackPtr]
  RETURNS[BODLEAN] =

BEGIN
  tempIndex: CARDINAL ← index;
  procCall: BOOLEAN ← FALSE;
  tempString: LONG STRING ← NIL;
  tempProcCount: CARDINAL ← 0;
  tempLineCount: CARDINAL ← 0;
  YesAProc: BOOLEAN ← FALSE;

  -- Examine procedure list to determine if there is a call match
  IF NOT procCall THEN IF ASP[index].idMark # procedureCall THEN
    {
      IF ASP[index].listMark # using THEN
        {
          WHILE tempProcCount < procListCount DO
            tempProcCount ← tempProcCount + 1;
            tempString ←
              String.CopyToNewString[ASP[procListPtr[tempProcCount]
                .index].inputString,z];
            IF String.Equal[tempString,ASP[index].inputString]
              THEN
                { -- Then check if the declaration has
                  parameters
                  -- if it does this is not a procedure call,
                  just
                  -- a data item that has the same name
                  tempLineCount ←
                    procListPtr[tempProcCount].index;
                  YesAProc ← TRUE;
                  WHILE ASP[tempLineCount].symbolNumber #
                    semiColon DO
                    tempLineCount ← tempLineCount + 1;
                    IF ASP[tempLineCount].symbolNumber =
                      openBracket
                      OR ASP[tempLineCount].symbolNumber =
                        returns THEN
                      {
                        YesAProc ← FALSE;
                        EXIT;
                      };
                    ENDLOOP; -- ASP
                  IF YesAProc THEN
                    {
                      ASP[index].idMark ← procedureCall;
                      procCall ← TRUE;
                      String.FreeString [z, tempString];
                      MarkCallList[index, ASP];
                      EXIT;
                    };
                  String.FreeString [z, tempString];
                }
              ENDLOOP; -- No parameter case
            };
          };
        }
      };
    }
  END;

  RETURN[procCall];
END; -- CheckIfLocalProcCall

--Check if 201 call parameter list
CheckIfCallList: PROCEDURE[index:CARDINAL, ASP:Parser.AtomStackPtr,
  fileStackPtr:Parser.FileStackPtr, fileCount:CARDINAL]
  procCount: CARDINAL, procStackPtr: ProcStackPtr]
  RETURNS[BOOLEAN] =

BEGIN
  tempIndex: CARDINAL ← index;
  callList: BOOLEAN ← FALSE;
  tempString: LONG STRING ← NIL;
  tempProcCount: CARDINAL ← 0;

  -- Examine procedure list to determine if there is a call match
  IF ASP[index - 1].idMark = procedureCall THEN
    {
      IF ASP[index].symbolNumber = openBracket THEN
        {
          callList ← TRUE;
          WHILE ASP[tempIndex].symbolNumber # closeBracket DO
            ASP[tempIndex].listMark ← callParameter;
            ASP[tempIndex].idMark ← callParameter;
            tempIndex ← tempIndex + 1;
          ENDLOOP;
        };
      };
    };
  RETURN[callList];
END; -- CheckIfCall

-- ListBreak looks for delimiters in lists to exit variable
checking
ListBreak: PROCEDURE [ASP: Parser.AtomStackPtr, count: CARDINAL]
  RETURNS [BOOLEAN] =
BEGIN
  listBreak: BOOLEAN ← FALSE;
  IF ASP[count].symbolNumber = closeBracket THEN listBreak ←
    TRUE;
  RETURN [listBreak];
END; -- Listbreak

-- MarkCallList: This procedure marks the call parameter list
MarkCallList: PROCEDURE[index:CARDINAL,ASP: Parser.AtomStackPtr] =
BEGIN
  -- Local Variables
  tempIndex: CARDINAL ← index + 1;

```



```

-- Procedure
WHILE asp[tempIndex].listMark = other DO
    asp[tempIndex].listMark ← callParameter;
    tempIndex ← tempIndex + 1;
ENDLOOP;
END; -- MarkCallList

-- Procedure parameter check. Returns true or false
ProcParameter: PROCEDURE [asp: Parser.AtomStackPtr, count:
CARDINAL] RETURNS [BOOLEAN] =
BEGIN
    procParameter: BOOLEAN ← FALSE;
    IF asp[count].listMark = parameter THEN procParameter ← TRUE;
    IF asp[count].listMark = return THEN procParameter ← TRUE;
    IF asp[count].listMark = returns THEN procParameter ← TRUE;
    RETURN [procParameter];
END; -- ProcParameter

-- Procedure checks for declarations of user declared types
CheckUserDeclType: PROCEDURE [index: CARDINAL, asp:
Parser.AtomStackPtr]
    RETURNS [found: BOOLEAN] =
BEGIN
    -- Local Variables
    count: CARDINAL ← 2;
    offset: CARDINAL ← index;
    loopCount: CARDINAL ← 0;
    allFound: BOOLEAN ← FALSE;

    -- Procedure start
    found ← FALSE;
    IF asp[index + 2].symbolNumber = public THEN {
        asp[index].varScope ← 'P';
        offset ← offset + 1;
    }

    WHILE NOT allFound DO
        IF asp[offset + 2].symbolNumber = 1 THEN {
            IF asp[offset + 3].symbolNumber # point
            THEN {
                asp[index].type ←
                    String.CopyToString[asp[offset + 2].inputString, z];
                asp[index].idMark ← variableDecl;
                asp[offset + 2].symbolNumber ← 0;
                found ← TRUE;
            }
            ELSE {
                asp[index].lEmark ← 'I';
                asp[index].idMark ← variableDecl;
                asp[index].type ←

```

```

String.CopyToString[asp[offset + 4].inputString, z];
asp[offset + 4].symbolNumber ← 0;
asp[index].lEmark ←
String.CopyToString[asp[offset + 2].inputString, z];
asp[offset + 2].symbolNumber ← 0;
found ← TRUE;};

-- Catch the case where multiple declarations are made
IF asp[index - 1].symbolNumber = semiColon THEN
{
    allFound ← TRUE;
    EXIT;
};
IF asp[index - 1].symbolNumber = comma THEN
{
    IF asp[index - 2].symbolNumber = 1 THEN
        index ← index - 2;
    };
    index ← index - 2;
}
ELSE EXIT; -- Get out from the loop no types found
ENDLOOP; -- allFound
IF allFound THEN found ← TRUE;
RETURN [found];
END; -- CheckUserDeclType

-- Procedure dereferences to find variable location
Dereference: PROCEDURE [index: CARDINAL, asp: Parser.AtomStackPtr]
=
BEGIN
    Imported: BOOLEAN ← FALSE;
    tempIndex: CARDINAL ← index;
    WHILE asp[tempIndex - 1].symbolNumber = point DO
        IF asp[tempIndex - 2].symbolNumber = closeBracket
        THEN {
            [tempIndex] ← SkipBrackets[index, tempIndex, asp]
        }
        ELSE {
            tempIndex ← tempIndex - 2;
            Imported ← TRUE;
        };
    ENDLOOP;
    CollectPathName[index, tempIndex, asp, Imported];
END; -- Dereference

-- Procedure checks USING clause for imported procedures
CheckUsingClause: PROCEDURE [index: CARDINAL, asp:
Parser.AtomStackPtr,
    fileStackPtr: Parser.FileStackPtr,
    fileCount: CARDINAL] =
BEGIN

```

```

tempIndex: CARDINAL ← fileStackPtr[fileCount].startIndex;
tempStmtCount: CARDINAL ← asp[tempIndex].sCount;

WHILE tempStmtCount = asp[tempIndex].sCount DO
  tempIndex ← tempIndex + 1;
  IF asp[index + 2].idMark = procedureCall
  THEN {
    IF String.Equal[asp[index +
      2].inputString,asp[tempIndex].inputString]
    THEN { -- Backup to find importing interface name
      WHILE asp[tempIndex].symbolNumber # using DO
        tempIndex ← tempIndex - 1;
      ENDOLOOP; -- Then store the value with the
        procedure
        asp[index + 2].IEatom ←
          String.CopyToNewString[asp[tempIndex -
            1].inputString,z];
        asp[index + 2].IEmark ← 'I;
        EXIT;
      };
    ENDOLOOP;
  };
END; -- Procedure CheckUsingClause

-- Procedure checks OPEN clause for imported procedures and
-- to understand if the file has already been parsed.
CheckOpenClause: PROCEDURE[index: CARDINAL, asp:
  Parser.AtomStackPtr,
  fileStackPtr:Parser.FileStackPtr,
  fileCount:CARDINAL] =
  BEGIN
    done: BOOLEAN ← FALSE;
    tempFileCount: CARDINAL ← 0;
    tempString: LONG STRING ← NIL;

    -- Attach .mesa extention to inputString
    tempString ←
      String.CopyToNewString[asp[index].inputString,z];
    String.AppendStringAndGrow[@tempString,".mesa"L,z];

    -- Tag idMark with the open clause identifier
    asp[index].idMark ← openClause;

    -- Check each fileName in turn to see if it is done
    WHILE fileCount > tempFileCount DO
      {
        tempFileCount ← tempFileCount + 1;
        IF
          String.Equal[tempString,fileStackPtr[tempFileCount].f
            ileName]
          THEN { done ← TRUE;
            EXIT; -- Compare file names with previously
              };
        ENDOLOOP;
      }
    END;
  
```

```

-- IF NOT done THEN {
--   Utils.log["\nOPEN clause, recommend you
process << "L];
--   Utils.log[asp[index].inputString];
--   Utils.log[" .mesa >> first."L];
--   String.FreeString[z,tempString];
--   END; -- Procedure CheckOpenClause

-- CollectPathName: This procedure collects the name of
-- a variable that has been dereferenced.
CollectPathName: PROCEDURE[index: CARDINAL, tempIndex:CARDINAL,
  asp: Parser.AtomStackPtr, Imported:BOOLEAN] =
  BEGIN
    asp[index].IEatom ←
      String.CopyToNewString[asp[tempIndex].inputString,z];
    IF Imported THEN asp[index].varScope ← 'M -- Imported case
      ELSE asp[index].varScope ← 'O; -- Full
        dereference
        asp[index].declLocation ← tempIndex;
        asp[tempIndex].symbolNumber ← 0;
        tempIndex ← tempIndex - 2;
      END; -- CollectPathName

SkipBrackets: PROCEDURE[index: CARDINAL, tempIndex: CARDINAL,
  asp:Parser.AtomStackPtr]
  RETURNS [CARDINAL] =
  BEGIN
    WHILE asp[tempIndex].symbolNumber # openBracket DO
      tempIndex ← tempIndex - 1;
    ENDOLOOP;
    tempIndex ← tempIndex - 1;
    RETURN[tempIndex];
  END; -- SkipBrackets

-- Pass4 Counts the procedure delarations and marks them,
Pass4: PUBLIC PROCEDURE [aIP:Parser.AtomIndexPtr, asp:
  Parser.AtomStackPtr,
  fSP:Parser.FileStackPtr, fileCount:CARDINAL] =
  BEGIN
    -- Local Variables
    loopCount:CARDINAL ← 0;
    tempString: LONG STRING ← NIL;
    -- Nesting variable
    -- Variables for the file level atomStack DataBase
  
```

```

NestStackPtr: TYPE = LONG POINTER TO NestStack;
NestStack: TYPE = ARRAY[0..MaxNest] OF NestProps;
NestProps: TYPE = RECORD [
    currentLevel: CARDINAL + 0,
    prevLevel: CARDINAL + 0];
MaxNest: CARDINAL = 30;

nSP: NestStackPtr + NIL;
nestCount: CARDINAL + 0;

-- Loop until the last file in the database has been parsed
loopCount + fSP[fileCount].startIndex;
nSP + z.NEW[NestStack];
IF fileCount > 1 THEN
    fSP[fileCount].procCount + fSP[fileCount - 1].procCount;

WHILE loopCount < aIP+ DO
    loopCount + loopCount + 1;
    -- Repeat current proc level in each cell
    aSP[loopCount].varMark + aSP[loopCount - 1].varMark;
    -- Look for a new procedure declaration
    IF aSP[loopCount].idMark = procedure THEN
        {
            nestCount + nestCount + 1;
            IF nestCount = MaxNest THEN
                {
                    Utils.log["\nError nesting level > "L];
                    tempString +
                    String.CopyToNewString[Utils.CardToString[MaxNest],z];
                    Utils.log[tempString];
                    String.freeString[z,tempString];
                    Utils.log["....\n\nL"];
                    nestCount + nestCount - 1;
                };
            fSP[fileCount].procCount + fSP[fileCount].procCount + 1;
            aSP[loopCount].varMark + fSP[fileCount].procCount;
            nSP[nestCount].currentLevel +
            fSP[fileCount].procCount;
            nSP[nestCount].prevLevel + aSP[loopCount - 1].varMark;
        };
    -- Look for the end of procedure declarations
    IF aSP[loopCount].pLevel < aSP[loopCount - 1].pLevel THEN
        {
            IF nestCount = 0 THEN
                -- End of procedure declaration
                {
                    aSP[loopCount].varMark + 0;
                }
            ELSE
                -- Catch the end of nested procedures
                {
                    aSP[loopCount].varMark +
                    nSP[nestCount].prevLevel;
                    nestCount + nestCount - 1;
                }
            }
        }
    }

```

```

};
ENDLOOP; -- loopCount
z.FREE[0nSP];
END; -- Pass4

-- Pass5 Check each variable and determines it's scope. The field
varScope is
-- marked as follows:
-- D = Dereference. This variable has a root already
specified
-- E = An External database
-- G = Global variable, ie confined to this impl
-- I = A variable that was instantiated (ie declared in
procedure header)
-- L = Local, confined to this procedure
-- M = Got to Start of File, therefore must be iMported
-- N = New zone declaration variable
-- P = Public variable
-- R = A return parameter, instantiated by a Call
Pass5: PUBLIC PROCEDURE [aIP:Parser.AtomIndexPtr, aSP:
Parser.AtomStackPtr,
fSP:Parser.FileStackPtr, fileCount:CARDINAL] =
BEGIN
    externalData: BOOLEAN + FALSE;
    loopCount: CARDINAL + 0;
    tempIndex: CARDINAL + 0;
    -- Loop until the last file in the database has been parsed
    loopCount + fSP[fileCount].startIndex;
    WHILE loopCount < aIP+ DO
        loopCount + loopCount + 1;
        -- Checking for declarations of external data sources
        IF aSP[loopCount].idMark = variableDec THEN
            {
                -- At this point a list should be examined of
                externally input cases for
                -- external data sources or sinks. This implementation
                simply hard codes
                -- in the known cases for external data, in this case
                MStream. If further
                -- cases are found they can be plugged in with an
                additional IF statement
                IF String.Equal[aSP[loopCount +
                2].inputString,"MStream"] THEN externalData + TRUE;
            }
            IF externalData THEN
                {
                    IF String.Equal[aSP[loopCount +
                    4].inputString,"Handle"] THEN

```

```

{
  ASP[loopCount].varScope ← 'E;
  externalData ← FALSE;
};
};

-- Checking for variables
IF ASP[loopCount].idMark = variable THEN
{
  -- Look for the declaration.
  -- However if this is a procedure formal parameter or
  -- returns parameter then the scope question is
  irrelevant.
  -- Also the scope of a Dereference variable is tied up
  -- with it's parent (eg ASP[index].varScope. varScope is
  -- a dereferenced variable, ASP is it's parent). Therefore
  -- only the parents are studied the 'D variable is
  bypassed.
  -- Variables in this category should be bypassed.
  IF ASP[loopCount].listMark # parameter AND
  ASP[loopCount].listMark # returns AND
  ASP[loopCount].varScope # 'D THEN
  {
    tempIndex ← loopCount;
    WHILE tempIndex > fSP[fileCount].startIndex DD
      tempIndex ← tempIndex - 1;

    -- First check to see if we are at beginning
    of file
    IF fSP[fileCount].startIndex = tempIndex THEN
    {
      IF ASP[loopCount].varScope = '-' THEN
      {
        ASP[loopCount].varScop
        e← 'M;
      };

      -- Second, check to see if this is a variable
      declaration
      IF ASP[tempIndex].idMark = variableDecl THEN
      {
        IF
          String.Equal[ASP[loopCount].inputStri
          ng, ASP[tempIndex].inputString] THEN
        {
          IF ASP[tempIndex].varScope =
          'P THEN
          {
            ASP[loopCount].varScop
            e← 'P;
            ASP[loopCount].declLoc
            ation ← tempIndex;
          }
          ELSE
          {
            IF

```

```

ASP[tempIndex].varScope = 'E THEN
{
  ASP[loopCount].va
  rScope ← 'E;
  ASP[loopCount].de
  clLocation ← tempIndex;
}
ELSE
{
  IF
    ASP[tempIndex].Plevel = 0 THEN
  {
    nt].varScope ← 'G;
    ASP[loopCou
    nt].declLocation ← tempIndex;
  };
  IF
    ASP[tempIndex].Plevel > 0 THEN
  {
    nt].varScope ← 'L;
    ASP[loopCou
    nt].declLocation ← tempIndex;
  };
};
EXIT;
};

-- Third, check if this is a variable declared
by the procedure parameter
IF ASP[tempIndex].listMark = parameter THEN
{
  IF
    String.Equal[ASP[loopCount].inputStri
    ng, ASP[tempIndex].inputString] THEN
  {
    -- But make sure this is the
    correct procedure, either at
    -- the same count level or a
    nested one.
    IF ASP[loopCount].varMark =
    ASP[tempIndex].varMark DR
    ASP[loopCount].Plevel >
    ASP[tempIndex].Plevel THEN
    {
      ASP[loopCount].varScope ←
      'I;
      ASP[loopCount].declLocati
      on ← tempIndex;
      EXIT;
    };
  };
};

-- Forth, check if this is a variable declared
by the return parameter
IF ASP[tempIndex].listMark = returns THEN

```

```

{
  IF
    String.Equal[asP[loopCount].inputString, asP[tempIndex].inputString] THEN
    { -- But make sure this is the
      correct procedure, either at
        -- the same count level or a
        nested one.
      IF asP[loopCount].varMark =
        asP[tempIndex].varMark OR
        asP[loopCount].Plevel >
        asP[tempIndex].Plevel THEN
        {
          asP[loopCount].varScope ←
            'R;
          asP[loopCount].declLocati
            on ← tempIndex;
          EXIT;
        };
      };
      -- Fifth, check if this is a variable within a
      new zone declaration
      IF asP[tempIndex].listMark = new THEN
      {
        asP[tempIndex].varScope ← 'N
      };

      ENOLOOP; -- tempIndex
    };

    ENOLOOP; -- loopCount

  END; -- Pass5

  -- Just a little straight line code
  -- Start up procedure list database
  BEGIN
    procListPtr ← z.NEW[ProcedureList];
    -- procStackPtr ← z.NEW[ProcedureStack];

  END;

END. -- End of ProgramParserImpl

```

```

-- File: RawDateImpl.mesa -- Last Edit:
-- EgertonWbst129:Xerox 13-Nov-93 10:37:28
--
*****
-- File: RawDataImpl.mesa
--
-- Oavid Egerton:Wbst129
--
-- Copyright (C) 1989 by Xerox Corporation. All rights reserved.
--
-- This module is responsible for the collection of data into
-- the output format and writing to the file DesignExtractor.data
--
*****

```

DIRECTORY

```

Ascii,
Display,
Environment,
Heap,
MFile,
MStream,
Stream,
String,
Parser,
Utils;

```

RawDataImpl: PROGRAM

```

IMPORTS Heap, MStream, Stream, String, Utils
EXPORTS Display =

```

BEGIN

```

  created: BOOLEAN ← TRUE;
  result: BOOLEAN ← FALSE;
  pageSize: CARDINAL ← 49;

  numString: LONG STRING ← NIL;
  z: UNCOUNTED_ZONE ← Heap.systemZone;

  statsFile: MStream.Handle;

```

-- Procedures

```

RawData: PUBLIC PROCEDURE [created: BOOLEAN, aStackPtr:
  Parser.AtomStackPtr,
  aIndexPtr: Parser.AtomIndexPtr, fileStackPtr:
  Parser.FileStackPtr,
  MaxFiles: CARDINAL, debugg: BOOLEAN] RETURNS [BOOLEAN] =

```

BEGIN

```

  -- Declare temporary variables
  result: BOOLEAN;

```

```

  -- Procedure starts here

```

```

  Utils.log["Creating output file .....\\n\\n"];

```

```

IF aIndexPtr > 0 THEN
  BEGIN
    Utils.CheckForAbort[];
    statsFile ← MStream.WriteOnly[
      "Extractor.data", [NIL, NIL].text];
    BEGIN
      ENABLE UNWIND => {
        IF statsFile#NIL THEN statsFile.delete[statsFile];
        statsFile ← NIL;
        FormatData[aIndexPtr, aStackPtr, fileStackPtr, MaxFiles];
        Utils.log["\\nDone ... Output created to file
          Extractor.data\\n"];
        Utils.log["\\n~ ~ ~ ~ ~ ~ ~ ~ ~ ~\\n"];
      };
    END;
    MStream.SetLength[statsFile,
      Stream.GetPosition[statsFile]];
    statsFile.delete[statsFile];
    statsFile ← NIL;
    result ← TRUE;
  END
ELSE
  Utils.log["Warning - No data in buffer, Extractor data
    file not created.....\\n\\n"];
END;
RETURN[result];
END; -- CreateFile

```

```

FormatData: PROCEDURE [aIP: LONG POINTER, atomStackPtr:
  Parser.AtomStackPtr, fileStackPtr: Parser.FileStackPtr, MaxFiles:
  CARDINAL] =

```

BEGIN

```

  -- Local variable declarations
  count, tempIndex: CARDINAL ← 0;
  maxAtomLength: CARDINAL ← 20;
  maxBEndLength: CARDINAL ← 4;
  maxDeclLocation: CARDINAL ← 4;
  maxIDmarkLength: CARDINAL ← 4;
  maxIEIdentifierLength: CARDINAL ← 15;
  maxIEopLength: CARDINAL ← 2;
  maxIndexLength: CARDINAL ← 4;
  maxLevelLength: CARDINAL ← 4;
  maxListMarkLength: CARDINAL ← 4;
  maxProcLevelLength: CARDINAL ← 4;
  maxRuleNumberLength: CARDINAL ← 4;
  maxStmntLength: CARDINAL ← 4;
  maxStringLength: CARDINAL ← 40;
  maxSymbolNumberLength: CARDINAL ← 4;
  maxTypeLength: CARDINAL ← 15;
  maxTypeBaseLength: CARDINAL ← 9;
  maxVarMarkLength: CARDINAL ← 4;
  maxVarScopeLength: CARDINAL ← 4;
  maxValuesLength: CARDINAL ← 20;

```

```

-- Write the Extraction data output file
WriteFiles[fileStackPtr, MaxFiles]; -- First page shows files parsed
WriteTitles[]; -- Set up the titles for the first page
WHILE tempIndex < aIP+ DO
    SendProgressMessage[tempIndex];
    Utils.CheckForAbort[];
    tempIndex + tempIndex + 1;

-- Write a title a the top of each page
IF tempIndex MOD pageSize = 0 THEN WriteTitles[];

-- Write out Index number value
numString ←
    String.CopyToNewString[Utils.CardToString[atomStackPtr[tempIndex]
        .Index],z];
CheckLength[maxIndexLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out Atom content
StatsWrite [" " "L"];
numString ←
    String.CopyToNewString[atomStackPtr[tempIndex].inputString,z];
CheckLength[maxAtomLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out Symbol number value
StatsWrite [" " "L"];
numString ←
    String.CopyToNewString[Utils.CardToString[atomStackPtr[tempIndex]
        .symbolNumber],z];
CheckLength[maxSymbolNumberLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out statement count number
StatsWrite [" " "L"];
numString ←
    String.CopyToNewString[Utils.CardToString[atomStackPtr[tempIndex]
        .sCount],z];
CheckLength[maxStmntLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out Begin End Level count
StatsWrite [" " "L"];
IF atomStackPtr[tempIndex].BElevel = 0 THEN {
    numString ← String.CopyToNewString[" " 0",z]}
ELSE {
    numString ←

```

```

    String.CopyToNewString[Utils.CardToString[atomStackPtr[tempIndex]
        x].BElevel],z});
CheckLength[maxBEEndLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out procedure indent level count
StatsWrite [" " "L"];
IF atomStackPtr[tempIndex].Plevel = 0 THEN {
    numString ← String.CopyToNewString[" " 0",z]}
ELSE {
    numString ←
        String.CopyToNewString[Utils.CardToString[atomStackPtr[tempIndex]
            dex].Plevel],z});
CheckLength[maxLevelLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out list mark
StatsWrite [" " "L"];
IF atomStackPtr[tempIndex].listMark = 0 THEN {
    numString ← String.CopyToNewString[" " 0",z]}
ELSE {
    numString ←
        String.CopyToNewString[Utils.CardToString[atomStackPtr[tempIndex]
            dex].listMark],z});
CheckLength[maxListMarkLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out id mark
StatsWrite [" " "L"];
IF atomStackPtr[tempIndex].idMark = 0 THEN {
    numString ← String.CopyToNewString[" " 0",z]}
ELSE {
    numString ←
        String.CopyToNewString[Utils.CardToString[atomStackPtr[tempIndex]
            dex].idMark],z});
CheckLength[maxIDmarkLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out variable mark
StatsWrite [" " "L"];
IF atomStackPtr[tempIndex].varMark = 0 THEN {
    numString ← String.CopyToNewString[" " 0",z]}
ELSE {
    numString ←
        String.CopyToNewString[Utils.CardToString[atomStackPtr[tempIndex]
            dex].varMark],z});
CheckLength[maxVarMarkLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out Variable scope designation
StatsWrite [" " "L"];
Stream.PutChar[statsfile,atomStackPtr[tempIndex].varScope];
StatsWrite[" " "L"];

```

```

-- Write out variable read or Write status
StatsWrite [" " "L"];
Stream.PutChar[statsFile,atomStackPtr[tempIndex].readWrite];
StatsWrite[" " "L"];

-- Write out Declaration Location
StatsWrite [" " "L"];
IF atomStackPtr[tempIndex].declLocation = 0
THEN numString ← String.CopyToString[" 0",z]
ELSE numString ←
    String.CopyToString[Utils.CardToString[atomStackPtr[temp
    Index].declLocation],z];
CheckLength[maxDeclLocation];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite[" " "L"];

-- Write out import export designation
Stream.PutChar[statsFile,atomStackPtr[tempIndex].IEmark];

-- Write out import export identifier
StatsWrite [" " "L"];
numString ←
    String.CopyToString[atomStackPtr[tempIndex].IEatom,z];
CheckLength[maxIdentifierLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out variable type designation
StatsWrite [" " "L"];
numString ←
    String.CopyToString[atomStackPtr[tempIndex].type,z];
CheckLength[maxTypeLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out variable values
StatsWrite [" " "L"];
numString ←
    String.CopyToString[atomStackPtr[tempIndex].values,z];
CheckLength[maxValuesLength];
StatsWrite [numString];
String.FreeString [z, numString];

Stream.PutChar [statsFile, Ascii.CR];
ENDLOOP; -- tempIndex

Stream.PutChar [statsFile, Ascii.CR];
Stream.PutChar [statsFile, Ascii.CR];
Stream.PutChar [statsFile, Ascii.CR];

END; -- FormatData

-- Writes the file information on the first page

```

```

WriteFiles: PROCEDURE [ fileStackPtr: Parser.FileStackPtr, MaxFiles:
CARDINAL] =
BEGIN
    -- Local variables
    maxFileIndexLength: CARDINAL ← 5;
    maxFileNameLength: CARDINAL ← 30;
    maxStartLength: CARDINAL ← 11;
    maxEndLength: CARDINAL ← 9;

    -- Procedure starts here
    tempIndex: CARDINAL ← 0;
    Stream.PutChar [statsFile, Ascii.CR];
    Stream.PutChar [statsFile, Ascii.CR];
    StatsWrite ["DESIGN EXTRACTION STATISTICS" "L"];
    Stream.PutChar [statsFile, Ascii.CR];
    StatsWrite ["===== " "L"];
    Stream.PutChar [statsFile, Ascii.CR];
    Stream.PutChar [statsFile, Ascii.CR];
    StatsWrite ["The files parsed in this session were:" "L"];
    Stream.PutChar [statsFile, Ascii.CR];
    Stream.PutChar [statsFile, Ascii.CR];
    StatsWrite [" Index File Name
    Index End Index" "L"];
    Stream.PutChar [statsFile, Ascii.CR];
    StatsWrite ["----- " "L"];
    Stream.PutChar [statsFile, Ascii.CR];

    -- Output file data
    WHILE tempIndex < MaxFiles DO
        tempIndex ← tempIndex + 1;
        IF fileStackPtr[tempIndex].fileCount = 0 THEN EXIT;

        -- Write out file count
        StatsWrite [" " "L"];
        numString ←
            String.CopyToString[Utils.CardToString[fileStackPtr[tempIn
            dex].fileCount],z];
        CheckLength[maxFileIndexLength];
        StatsWrite [numString];
        String.FreeString [z, numString];

        -- Write out file name
        StatsWrite [" " "L"];
        numString ←
            String.CopyToString[fileStackPtr[tempIndex].fileName,z];
        CheckLength[maxFileNameLength];
        StatsWrite [numString];
        String.FreeString [z, numString];

        -- Write out file start index
        StatsWrite [" " "L"];
        numString ←
            String.CopyToString[Utils.CardToString[fileStackPtr[tempIn
            dex].startIndex],z];
        CheckLength[maxStartLength];
        StatsWrite [numString];
    
```



```

String.FreeString [z, numString];

-- Write out file end index
StatsWrite [" " "L"];
numString ←
String.CopyToNewString[Utils.CardToString[fileStackPtr[tempIn
dex].endIndex],z];
CheckLength[maxEndLength];
StatsWrite [numString];
String.FreeString [z, numString];

Stream.PutChar [statsFile, Ascii.CR];

ENDLOOP; -- File output

-- Complete page
WHILE tempIndex <= (pageSize) DO
tempIndex ← tempIndex + 1;
Stream.PutChar [statsFile, Ascii.CR];
ENDLOOP; -- Complete Page

END; -- WriteFiles

```

-- Writes the title information at the top of each page

```

WriteTitles: PROCEDURE =
BEGIN
Stream.PutChar [statsFile, Ascii.CR];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["DESIGN EXTRACTION STATISTICS" L];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["===== L"];
Stream.PutChar [statsFile, Ascii.CR];
Stream.PutChar [statsFile, Ascii.CR];

StatsWrite ["Ind Atom Symb Stmt BEND Proc List
id Var Read Decl Import.....Export
Values " L];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["# Description # Cnt lvl lvl mark
mark mark Scope Write Loca Op Identifier" L];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["----- L]
----- L];
Stream.PutChar [statsFile, Ascii.CR];
Stream.PutChar [statsFile, Ascii.CR];
END; -- WriteTitles

```

-- Writes a string to the statistics output file

```

StatsWrite: PROCEDURE [str: LONG STRING] =
BEGIN
Stream.PutString [statsFile, str];
END; -- StatsWrite

-- Checks the length of the string and adds in spaces if short
CheckLength: PROCEDURE [length: CARDINAL] =

BEGIN
-- Local variable
size: CARDINAL;

-- Procedure body
size ← String.Length[numString];
IF size < length THEN AddSpaces[size,length];
IF size > length THEN ReduceLength[size,length];
END; -- CheckLength;

AddSpaces: PROCEDURE[size: CARDINAL, length: CARDINAL] =

BEGIN
size ← length - size; -- Determine how many spaces need to be
added
WHILE size > 0 DO
String.AppendStringAndGrow[@numString, ,z];
size ← size - 1;
ENDLOOP;

END; -- AddSpaces

ReduceLength: PROCEDURE[size: CARDINAL, length: CARDINAL] =

BEGIN
-- Substring allows the extraction of a portion of a string
subStringDescr: String.SubStringDescriptor;
tempString: LONG STRING ← NIL;

-- Procedure body
subStringDescr ← [numString,0,length];
tempString ← String.MakeString[z,length];

String.AppendSubString[tempString,@subStringDescr];
String.FreeString [z, numString];
numString ← String.CopyToNewString[tempString,z];

END; -- ReduceLength

SendProgressMessage: PROCEDURE [ti: CARDINAL] =

```

```
BEGIN
count: LONG STRING;

If tI = 0 THEN Utils.log["\nLines processed ...  "L];

If tI MOD 100 = 0 THEN {
    count ← Utils.CardToString[tI];
    Utils.log[count];
    Utils.log["L"];
};

END; -- SendProgressMessage

END. -- RawDataImp1
```

```

-- EgertonWbst129:Xerox 13-Nov-93 10:48:28
-- *****
-- File: StructChartImpl.mesa
-- David Egerton:Wbst129
-- Copyright (C) 1989 - 1993 by Xerox Corporation. All rights
reserved.
-- This module is responsible for the collection of data into
-- the output format and writing to the file StructChart.data
-- *****

DIRECTORY
  Ascii,
  Display,
  Environment,
  Heap,
  MFile,
  MStream,
  Stream,
  String,
  Parser,
  Utils;

StructChartImpl: PROGRAM
  IMPORTS Display, Heap, MStream, Stream, String, Utils
  EXPORTS Display =

BEGIN
  -- Variables to maintain position information as the program
  -- searches through the files for procedures
  PositionStackPtr: TYPE = LONG POINTER TO PositionStack;
  PositionStack: TYPE = ARRAY[0..maxLevels] OF PositionStack;
  PositionProps: TYPE = RECORD [
    callLocation: CARDINAL ← 0,
    declLocation: CARDINAL ← 0,
    procLevel: CARDINAL ← 0,
    dprocLevel: CARDINAL ← 0];
  maxLevels: CARDINAL = 40;

  -- Temporary stack to log missing export information
  -- The stack is used to look for duplication prior to output
  ExportStackPtr: TYPE = LONG POINTER TO ExportStack;
  ExportStack: TYPE = ARRAY[0..maxExportStack] OF ExportStackProps;
  ExportStackProps: TYPE = RECORD [
    missingProcLocation: CARDINAL ← 0];
  maxExportStack: CARDINAL = Parser.MaxString/400;

  -- Variable declarations
  fullCount: CARDINAL ← 0;

  -- Miscellaneous variables

  created: BOOLEAN ← TRUE;
  result: BOOLEAN ← FALSE;
  pageSize: CARDINAL ← 58;

  -- Formatting variables
  maxAtomLength: CARDINAL ← 20;
  maxBEndLength: CARDINAL ← 4;
  maxFileLength: CARDINAL ← 20;
  maxIDmarkLength: CARDINAL ← 4;
  maxIEIdentLength: CARDINAL ← 15;
  maxIEopLength: CARDINAL ← 2;
  maxIndexLength: CARDINAL ← 4;
  maxLevelLength: CARDINAL ← 4;
  maxListMarkLength: CARDINAL ← 4;
  maxLVLlength: CARDINAL ← 3;
  maxProcLevelLength: CARDINAL ← 20;
  maxRepeatLength: CARDINAL ← 4;
  maxRuleNumberLength: CARDINAL ← 4;
  maxStmntLength: CARDINAL ← 4;
  maxStringLength: CARDINAL ← 40;
  maxSubMargin: CARDINAL ← 74;
  maxSymbolNumberLength: CARDINAL ← 4;
  maxTypeLength: CARDINAL ← 15;
  maxTypeBaseLength: CARDINAL ← 9;
  maxVarMarkLength: CARDINAL ← 4;
  maxVarScopeLength: CARDINAL ← 4;
  maxValuesLength: CARDINAL ← 25;

  -- Lexical variables
  equals: CARDINAL ← 15;
  import: CARDINAL ← 65;
  monitor: CARDINAL ← 44;
  other: CARDINAL ← 201;
  callParameter: CARDINAL ← 238;
  parameter: CARDINAL ← 200;
  program: CARDINAL ← 43;
  procCall: CARDINAL ← 237;
  procedure: CARDINAL ← 37;
  semiColon: CARDINAL ← 10;
  variable: CARDINAL ← 500;

  numString: LONG STRING ← NIL;

  -- Setup variables
  z: UNCOUNTED ZONE ← Heap.systemZone;
  statsFile: MStream.Handle;
  positionStackPtr: PositionStackPtr ← NIL;
  fullStackPtr: Display.FullStackPtr ← NIL;
  exportStackPtr: ExportStackPtr ← NIL;
  stackIndex: CARDINAL ← 0;
  posIndex: CARDINAL ← 0;
  lineCount: CARDINAL ← 0;

```



```

posIndex ← posIndex + 1;
Utils.CheckForAbort[];

CheckReturnCondition[atomStackPtr, debugg];

SELECT atomStackPtr[posIndex].idMark FROM

37 =>
  BEGIN -- idMark is a procedure declaration
  IF stackIndex = 0 THEN
    {
      -- Ist stackIndex procedure mark it
      IF CheckIfTerminalDecl[atomStackPtr, fileStackPtr]
      THEN
        { -- Terminal declaration, log it and write
          it out
          SLC ← TRUE;
          stackIndex ← stackIndex + 1; -- increase
          stackIndex number
          CleanUpPositionData[];
          CollectPositionData[atomStackPtr, debugg];
          UpdateFullStack[atomStackPtr, fileStackPtr,
            SLC, found];
          SLC ← FALSE;
        }
      } ELSE
        { -- Non terminal so suppress output
          SuppressOutput[atomStackPtr, debugg];
        };
      }
    } ELSE
      {
        -- Nested, non called declaration
        SuppressOutput[atomStackPtr, debugg];
      };
    }
  END;

237 =>
  BEGIN -- idMark is a procedure call
  IF stackIndex = 0 THEN SLC ← TRUE;
  stackIndex ← stackIndex + 1; -- increase stackIndex
  number
  CleanUpPositionData[];
  CollectPositionData[atomStackPtr, debugg];
  UpdateFullStack[atomStackPtr, fileStackPtr, SLC,
    found];
  SLC ← FALSE;
  [found] ←
  SearchForDeclaration[atomStackPtr, fileStackPtr,
    MaxFiles];
  IF found THEN
    {
      CollectPositionData[atomStackPtr, debugg];
      UpdateFullStack[atomStackPtr, fileStackPtr,
        SLC, found];
      GoFindCallParams[aIP, atomStackPtr, fullStackP
        tr];
    }
  } ELSE stackIndex ← stackIndex - 1;

```

END;

```

ENDCASE;
found ← FALSE;
IF posIndex > aIP↑ THEN
  {
    posIndex ← 0;
    EXIT;
  };
ENDLOOP; -- tempIndex

Utils.log["      - Studying repeat calls.....\n"];
StudyRepeats[atomStackPtr];
Utils.log["      - Creating Semi Graphical Output.....\n"];
IF SCGraph THEN
  {
    WriteSCTitles[]; -- Set up the titles for the Structure Chart
    SemiGraphOutput[atomStackPtr, fileStackPtr, debugg, repeats];
    Utils.log["      - Creating Tabular Output.....\n"];
  };
  IF SCTable THEN TabularOutput[atomStackPtr, fileStackPtr, debugg,
    repeats];
END; -- FormatData

-- CleanUpPositionData: This procedure makes sure the stack is
empty
-- prior to update. This ensures data is always the latest thus
-- avoids making decisions on data left from a previous state.
CleanUpPositionData: PROCEDURE =
  BEGIN
    positionStackPtr[stackIndex].declLocation ← 0;
    positionStackPtr[stackIndex].dProcLevel ← 0;
    positionStackPtr[stackIndex].callLocation ← 0;
    positionStackPtr[stackIndex].procLevel ← 0;

  END; -- CleanUpPositionData

-- CheckIfTerminalDecl: This procedure is called only when a
declaration
-- is found at the zero level. If this declaration is called
from
-- within the control file then the declaration should be ignored
-- at this point. It will be picked up later by the call.
CheckIfTerminalDecl: PROCEDURE[ASP:Parser.AtomStackPtr,
  fSP:Parser.FileStackPtr] RETURNS[BOOLEAN] =
  BEGIN
    tempIndex: CARDINAL ← 0;
    tempFileCount: CARDINAL ← 1;

```

```

terminal: BOOLEAN ← TRUE;

WHILE tempIndex < fSP[tempFileCount].endIndex DO
  tempIndex ← tempIndex + 1;
  IF aSP[tempIndex].idMark = procCall THEN
    {
      IF
        String.Equal[aSP[posIndex].inputString,aSP[tempIndex].in
        putString]
      THEN
        {
          terminal ← FALSE;
          EXIT;
        };
    };
  ENDLOOP; -- tempIndex
RETURN[terminal];
END; -- CheckIfTerminalDecl

```

```

-- CollectPositionData: Creates a temporary file that holds the
-- data required to understand when a stackIndex changed and most
-- importantly retrace steps when required.

```

```

CollectPositionData: PROCEDURE [aSP:
  Parser.AtomStackPtr,debugg:BOOLEAN] =
BEGIN
  IF aSP[posIndex].idMark = procedure THEN
    {
      positionStackPtr[stackIndex].declLocation ← posIndex;
      positionStackPtr[stackIndex].dProcLevel ←
      aSP[posIndex].Plevel;
    }
  ELSE
    {
      positionStackPtr[stackIndex].callLocation ← posIndex;
      positionStackPtr[stackIndex].procLevel ← aSP[posIndex].Plevel;
    };
  };

```

```

-- Debugg information only
IF debugg THEN

```

```

{
  StatsWrite["Call Location = "L];
  numString ←
  String.CopyToNewString[Utils.CardToString[positionStackPtr
  [stackIndex].callLocation],z];
  StatsWrite [numString];
  String.FreeString [z, numString];
  StatsWrite[" Proc Level = "L];
  numString ←
  String.CopyToNewString[Utils.CardToString[positionStackPtr
  [stackIndex].procLevel],z];
  StatsWrite [numString];
  String.FreeString [z, numString];
  StatsWrite[" Decl Location = "L];
  numString ←
  String.CopyToNewString[Utils.CardToString[positionStackPtr

```

```

[stackIndex].declLocation],z];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite[" Decl proc Level = "L];
numString ←
String.CopyToNewString[Utils.CardToString[positionStackPtr
[stackIndex].dProcLevel],z];
StatsWrite [numString];
String.FreeString [z, numString];
Stream.PutChar [StatsFile, Ascii.CR];
};
END; -- CollectPositionData

-- SemiGraphOutput: Sends data out to the structure chart file
-- in a semi graphical form
SemiGraphOutput: PROCEDURE [aSP: Parser.AtomStackPtr,
  fSP: Parser.FileStackPtr,
  debugg: BOOLEAN, repeats:BOOLEAN] =

```

```

BEGIN

```

```

-- Local variables
deref: CHAR;
pi: CARDINAL ← 0;
tempLevel: CARDINAL ← 0;
tempCount: CARDINAL ← 0;
fileNumber: CARDINAL ← 0;
currentCall: CARDINAL ← 0;
currentDecl: CARDINAL ← 0;
lastCall: CARDINAL ← 0;
currentLevel: CARDINAL ← 0;
SLC: BOOLEAN ← FALSE;

```

```

-- Loop until all the full Stack database has been completed
WHILE tempCount < fullCount DO
  tempCount ← tempCount + 1;
  tempLevel ← 1;

```

```

-- Suppress output if the procedure call is a repeat at this
  level
  currentCall ← fullStackPtr[tempCount].callLocation;
  currentLevel ← fullStackPtr[tempCount].callLevel;
  SLC ← fullStackPtr[tempCount].SLC;

```

```

  IF NOT fullStackPtr[tempCount].twinRepeat OR repeats THEN
    {

```

```

      -- Write out line index number
      numString ←
      String.CopyToNewString[Utils.CardToString[tempCount],z];
      CheckLength[maxIndexLength];
      StatsWrite [numString];
      String.FreeString [z, numString];
      StatsWrite [" "L];
    }
  }

```

```

-- Write file name that procedure is called from
fileName ← fullStackPtr[tempCount].fileNumber;
numString ←
String.CopyToNewString[fSP[fileNumber].fileName,z];
CheckLength[maxStringLength];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" " "L"];

-- Write out Importing interface name
numString ←
String.CopyToNewString[aSP[currentCall].iEatom,z];
CheckLength[maxIdentLength];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" " "L"];

-- Write file name that procedure is declared in
currentDecl ← aSP[currentCall].declLocation;
deref ← aSP[currentCall].varScope;
IF currentDecl = 0 OR deref = 'M THEN
{
  numString ← String.CopyToNewString["- ",z];
}
ELSE
{
  fileNumber ← FindFileNumber[fSP,currentDecl];
  numString ←
String.CopyToNewString[fSP[fileNumber].fileName,z];
};
CheckLength[maxStringLength];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" " "L"];

-- Write out procedure stackIndex number
numString ←
String.CopyToNewString[Utils.CardToString[currentLevel],z];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" " "L"];

-- Write out SLC status
IF SLC THEN StatsWrite["SLC "]
ELSE StatsWrite [" " "L"]; -- Jump over SLC notation

-- Write out procedure name multiplied by correct
stackIndex
WHILE tempLevel < currentLevel 00
  StatsWrite[" " "L"];
tempLevel ← tempLevel + 1;
ENLOOP; -- TempstackIndex
numString ←
String.CopyToNewString[aSP[currentCall].inputString,z];
StatsWrite [numString];
String.FreeString [z, numString];

```

```

Stream.PutChar [statsFile, Ascii.CR];
lineCount ← lineCount + 1;

-- Complete page
IF lineCount MOD pageSize = 0 THEN
{
  WritesCTitles[];
};
}
ELSE -- Skip to next call at this level as this is a
consecutive
-- repeat call of the procedure
{
  -- Move forward until same level
  WHILE currentLevel < fullStackPtr[tempCount + 1].callLevel 00
  IF tempCount > fullCount THEN EXIT;
  tempCount ← tempCount + 1;
  ENLOOP; -- currentLevel
};
ENLOOP; -- tempCount

-- Complete page
WHILE lineCount MOD pageSize # 0 00
  lineCount ← lineCount + 1;
Stream.PutChar [statsFile, Ascii.CR];
ENLOOP; -- Complete Page

END; -- SemiGraphOutput

-- UpdateFullStack: Each time an output is given for the
-- semi graphical representation the full stack database
-- is updated for later processing of a tabular output
UpdateFullStack: PROCEDURE[aSP: Parser.AtomStackPtr, fSP:
Parser.FileStackPtr,
                                SLC: BOOLEAN, found:BOOLEAN] =
BEGIN
-- Variables
count: LONG STRING ← NIL;
fileNumber: CAROINAL ← 0;

-- Procedure
-- Error if greater than max
IF fullCount >= Display.maxStack THEN
{
  count ← Utils.CardToString[Display.maxStack];
  Utils.log["Error ... fullStackCnt is >..."L];
  Utils.log[count];
  Utils.log["\n\n"L];
  fullCount ← fullCount - 1;
};
IF found THEN
{
  fullStackPtr[fullCount].declLocation ←
  positionStackPtr[stackIndex].declLocation;
}

```

```

}
ELSE
{
  fullCount ← fullCount + 1;
  fileNumber ← FindFileNumber[fSP, posIndex];
  fullStackPtr[fullCount].callLocation ← posIndex;
  fullStackPtr[fullCount].callLevel ← stackIndex;
  fullStackPtr[fullCount].fileNumber ← fileNumber;
  IF SLC THEN
    fullStackPtr[fullCount].SLC ← TRUE;
  fullStackPtr[fullCount].declLocation ← posIndex;
};
END; -- UpdateFullStack

-- GoFindCallParameters: This procedure adds to the atomStack
-- the declaration location of
-- each of the call parameters used in the procedure call.
GoFindCallParams: PROCEDURE
[aIP:Parser.AtomIndexPtr, aSP:Parser.AtomStackPtr,
fullStackPtr:Display.FullStackPtr] =
BEGIN
  -- Local Variables
  callIndex: CARDINAL ← 0;
  declIndex: CARDINAL ← 0;
  declParamCount: CARDINAL ← 0;
  paramCount: CARDINAL ← 0;
  tempLevel: CARDINAL ← 0;

  callIndex ← fullStackPtr[fullCount].callLocation;
  -- Move forward to beginning of parameter list
  WHILE aSP[callIndex].listMark # callParameter DO
    callIndex ← callIndex + 1;
  IF aSP[callIndex].symbolNumber = semiColon THEN EXIT;
  IF aSP[callIndex].symbolNumber = equals THEN EXIT;
  ENLOOP; -- aSP

  -- Count through the parameter list to see how many there are
  WHILE aSP[callIndex].listMark = callParameter DO
    callIndex ← callIndex + 1;
  -- Pick out the variables
  IF aSP[callIndex].symbolNumber = semiColon THEN EXIT;
  IF aSP[callIndex - 1].symbolNumber = equals THEN EXIT;
  IF aSP[callIndex].idMark = variable THEN
    {
      paramCount ← paramCount + 1;

      -- Go pick up the declaration location for this parameter
      -- Move forward to beginning of call parameter list
      declIndex ← fullStackPtr[fullCount].declLocation;
      declParamCount ← 0;
      WHILE aSP[declIndex].listMark # parameter DO
        declIndex ← declIndex + 1;
      IF aSP[declIndex].symbolNumber = semiColon THEN
        EXIT;
      ENLOOP; -- aSP
    }
  }

  -- Look for paramCount match
  WHILE aSP[declIndex].listMark = parameter DO
    declIndex ← declIndex + 1;
  IF aSP[declIndex].symbolNumber = semiColon THEN
    EXIT;
  IF aSP[declIndex].idMark = variable THEN
    {
      declParamCount ← declParamCount + 1;
      IF declParamCount = paramCount THEN
        {
          aSP[callIndex].declLocation ←
            declIndex;
          EXIT;
        };
      ENLOOP; -- declParamCount
    };
  ENLOOP; -- aSP
END; -- GoFindCallParams

-- SuppressOutput: This procedure passes over a procedure
-- declaration that is being parsed if the Structure Chart
-- stackIndex is greater than one. This is required because the
-- program is determining dynamic not static information.
-- Returns the new position to continue parsing. ie the
-- location of the end of the suppressed procedure.
SuppressOutput: PROCEDURE [aSP:
Parser.AtomStackPtr, debugg:BOOLEAN] =
BEGIN
  tempIndex:
  startingPLevel: CARDINAL ← posIndex;
  startingPLevel: CARDINAL;

  startingPLevel ← aSP[tempIndex].Plevel;
  IF debugg THEN
    {
      -- Write out location Index number value
      StatsWrite ["SuppressOutput start = "L];
      numString ←
        String.CopyToNewString[Utils.CardToString[posIndex], z];
      CheckLength[maxIndexLength];
      StatsWrite [numString];
      String.FreeString [z, numString];
      StatsWrite [" "L];
    };
  WHILE aSP[tempIndex].Plevel >= startingPlevel DO
    tempIndex ← tempIndex + 1;
  ENLOOP; -- tempIndex
  posIndex ← tempIndex;

```



```

IF debugg THEN
{
  -- Write out location Index number value
  StatsWrite ["SuppressOutput end = "L];
  numString ←
    String.CopyToNewString[Utils.CardToString[posIndex],z];
  CheckLength[maxIndexLength];
  StatsWrite [numString];
  String.FreeString [z, numString];
  Stream.PutChar [statsFile, Ascii.CR];
};

ENO; -- Suppress procedure

-- SearchForDeclaration: This procedure is called when a
-- procedure call is found. It goes off to find the declaration
-- and returns the new positional information. If not found the
boolean
-- remains false.
SearchForDeclaration: PROCEDURE [aSP: Parser.AtomStackPtr, fSP:
Parser.FileStackPtr, MaxFiles:CAROINAL]
  RETURNS [BOOLEAN] =
BEGIN
  callLocation: CAROINAL ← 0;
  found: BOOLEAN ← FALSE;
  tempFileCount:CAROINAL ← 0;
  tempIndex: CAROINAL ← 0;
  OK: BOOLEAN ← FALSE;
  -- Only process items with a valid import entry
  tempFileCount ← 0;
  WHILE tempFileCount < MaxFiles DO
    tempFileCount ← tempFileCount + 1;
    IF fSP[tempFileCount].fileCount = 0 THEN EXIT;

    -- Check if there is a matching export
    IF String.Equal[aSP[posIndex].lAtom,
fSP[tempFileCount].exportName]
    THEN BEGIN
      numString ←
        String.CopyToNewString[aSP[posIndex].inputString,z
];
      [found] ← CheckProcExists[aSP, fSP, tempFileCount,
numString];
      String.FreeString[z,numString];
      IF found THEN EXIT; -- Export information
      available
    ENO;
  ENOLOOP; -- tempFileCount

  -- Look locally for the procedure declaration
  callLocation ← aSP[posIndex].Index;
  tempFileCount ← FindFileNumber[fSP,callLocation];

```

```

};
};
}
ELSE
{
-- Decide if this is a first level situation. If so the
-- declarations are starter nodes, stackIndex = 1.
IF stackIndex = 1 THEN
{
IF positionStackPtr[stackIndex].dProcLevel >
aSP[posIndex].Plevel
THEN { stackIndex + stackIndex - 1;

IF debugg THEN
{
-- Write out location Index number value
StatsWrite ["End of a starter node = "L];
numString ←
String.CopyToString[Utils.CardToString[posIndex],
z];
CheckLength[maxIndexLength];
StatsWrite [numString];
String.FreeString [z, numString];
Stream.PutChar [statsFile, Ascii.CR];
};
}
}
ELSE -- When stackIndex is greater than 1
{
IF positionStackPtr[stackIndex].dProcLevel >
aSP[posIndex].Plevel
THEN {
posIndex ←
positionStackPtr[stackIndex].callLocation;
stackIndex ← stackIndex - 1;
posIndex ← posIndex + 1;

IF debugg THEN
{
-- Write out location Index number value
StatsWrite ["End of a starter node = "L];
numString ←
String.CopyToString[Utils.CardToString[posIndex],z];
CheckLength[maxIndexLength];
StatsWrite [numString];
String.FreeString [z, numString];
Stream.PutChar [statsFile, Ascii.CR];
};
}
};
};
END; -- CheckReturnCondition

```

```

-- FindFileNumber: This procedure returns for the file number
-- within which the current position index is pointing.
FindFileNumber: PROCEDURE [fSP: Parser.FileStackPtr,
callLocation: CARDINAL]
    RETURNS [CARDINAL] =
BEGIN
    fileCount: CARDINAL ← 1;
    notFound: BDDLEAN ← TRUE;

    WHILE notFound DD
        IF callLocation > fSP[fileCount].startIndex AND
        callLocation < fSP[fileCount].endIndex THEN
        {
            notFound ← FALSE;
            EXIT;
        };

        fileCount ← fileCount + 1;

    ENDDDDP; -- callLocation

    RETURN[fileCount];
END; -- FindFileNumber

-- StudyRepeats: This procedure looks at the fullStack
-- database and marks the repeat conditions that can
-- occur. This allows us to suppress repeats when
-- creating an output file. Just to make the output
-- more readable.
StudyRepeats: PROCEDURE [aSP: Parser.AtomStackPtr] =
BEGIN
    -- Local variables
    mainLoopCount: CARDINAL ← 1;
    localLoopCount: CARDINAL ← 1;
    currentCall: CARDINAL ← 1;
    comparisonCall: CARDINAL ← 1;
    currentLevel: CARDINAL ← 1;
    comparisonLevel: CARDINAL ← 1;

    -- Procedure
    WHILE mainLoopCount < fullCount DD
        mainLoopCount ← mainLoopCount + 1;
        currentCall ← fullStackPtr[mainLoopCount].callLocation;
        currentLevel ← fullStackPtr[mainLoopCount].callLevel;

        -- Check IF twin repeat situation, backup to last call
        -- at this level
        localLoopCount ← mainLoopCount;

        -- StudyRepeats: This procedure looks at the fullStack
        -- database and marks the repeat conditions that can
        -- occur. This allows us to suppress repeats when
        -- creating an output file. Just to make the output
        -- more readable.
        StudyRepeats: PROCEDURE [aSP: Parser.AtomStackPtr] =
        BEGIN
            -- Local variables
            mainLoopCount: CARDINAL ← 1;
            localLoopCount: CARDINAL ← 1;
            currentCall: CARDINAL ← 1;
            comparisonCall: CARDINAL ← 1;
            currentLevel: CARDINAL ← 1;
            comparisonLevel: CARDINAL ← 1;

            -- Procedure
            WHILE mainLoopCount < fullCount DD
                mainLoopCount ← mainLoopCount + 1;
                currentCall ← fullStackPtr[mainLoopCount].callLocation;
                currentLevel ← fullStackPtr[mainLoopCount].callLevel;

                -- Check IF twin repeat situation, backup to last call
                -- at this level
                localLoopCount ← mainLoopCount;

```

```

WHILE localLoopCount > 0 DO
  localLoopCount ← localLoopCount - 1;
  IF fullStackPtr[localLoopCount].callLevel < currentLevel THEN
    EXIT;
  IF fullStackPtr[localLoopCount].callLevel = currentLevel THEN
    {
      comparisonCall ←
        fullStackPtr[localLoopCount].callLocation;
      IF String.Equal[asP[currentCall].inputString,
        asP[comparisonCall].inputString] THEN
        {
          fullStackPtr[mainLoopCount].twinRepeat ← TRUE;
          fullStackPtr[localLoopCount].firstDfTwin ←
            TRUE;
          EXIT;
        };
      };
    ENDOODP; -- localLoopCount;

    -- Check for Global & Level repeats
    localLoopCount ← 0;

    WHILE localLoopCount < (mainLoopCount - 1) DO
      localLoopCount ← localLoopCount + 1;
      comparisonCall ←
        fullStackPtr[localLoopCount].callLocation;
      comparisonLevel ← fullStackPtr[localLoopCount].callLevel;
      IF String.Equal[asP[currentCall].inputString,
        asP[comparisonCall].inputString] THEN
        {
          fullStackPtr[mainLoopCount].globalRepeat ← TRUE;
          fullStackPtr[localLoopCount].firstDfGlobal ←
            TRUE;
          IF comparisonLevel = currentLevel THEN
            {
              fullStackPtr[mainLoopCount].levelRepeat ←
                TRUE;
              fullStackPtr[localLoopCount].firstDfLevel ←
                TRUE;
              EXIT;
            };
          };
        ENDOODP; -- global repeat check

      ENDOODP; -- main Loop

    END; -- StudyRepeats

```

-- Writes the file information on the first page

```

WriteFiles: PROCEDURE [ fileStackPtr: Parser.FileStackPtr, MaxFiles:
  CARDINAL ] =
  BEGIN
    -- Local variables
    maxFileIndexLength: CARDINAL ← 5;
    maxFileNameLength: CARDINAL ← 30;
    maxStartLength: CARDINAL ← 11;
    maxEndLength: CARDINAL ← 9;
    maxExportNameLength: CARDINAL ← 25;

    -- Procedure starts here
    tempIndex: CARDINAL ← 0;
    titleSize: CARDINAL ← 6;

    StatsWrite ["DESIGN EXTRACTION STATISTICS"L];
    Stream.PutChar [statsFile, Ascii.CR];
    StatsWrite ["===== "L];
    Stream.PutChar [statsFile, Ascii.CR];
    Stream.PutChar [statsFile, Ascii.CR];
    StatsWrite ["The files parsed in this session were:"L];
    Stream.PutChar [statsFile, Ascii.CR];
    Stream.PutChar [statsFile, Ascii.CR];
    StatsWrite [" Index File Name Start
    Index Exports to"L];
    Stream.PutChar [statsFile, Ascii.CR];
    StatsWrite ["----- "L];
    Stream.PutChar [statsFile, Ascii.CR];

    -- Update the line count
    lineCount ← lineCount + titleSize;

    -- Output file data
    WHILE tempIndex < MaxFiles DO
      tempIndex ← tempIndex + 1;
      lineCount ← lineCount + 1;
      IF fileStackPtr[tempIndex].fileCount = 0 THEN EXIT;

      -- Write out file count
      StatsWrite [" "L];
      numString ←
        String.CopyToNewString[Utils.CardToString[fileStackPtr[tempIn
          dex].fileCount],z];
      CheckLength[maxFileIndexLength];
      StatsWrite [numString];
      String.FreeString [z, numString];

      -- Write out file name
      StatsWrite [" "L];
      numString ←
        String.CopyToNewString[fileStackPtr[tempIndex].fileName,z];
      CheckLength[maxFileNameLength];
      StatsWrite [numString];
      String.FreeString [z, numString];
    
```

```

-- Write out file start index
StatsWrite [" " "L"];
numString ←
String.CopyToNewString[Utils.CardToString[fileStackPtr[tempIn
dex].startIndex,z];
CheckLength[maxStringLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out file end index
StatsWrite [" " "L"];
numString ←
String.CopyToNewString[Utils.CardToString[fileStackPtr[tempIn
dex].endIndex,z];
CheckLength[maxStringLength];
StatsWrite [numString];
String.FreeString [z, numString];

-- Write out export information
StatsWrite [" " "L"];
numString ←
String.CopyToNewString[fileStackPtr[tempIndex].exportName,z];
CheckLength[maxExportNameLength];
StatsWrite [numString];
String.FreeString [z, numString];

Stream.PutChar [statsFile, Ascii.CR];

ENDLOOP; -- File output

-- Complete page
WHILE lineCount MOD pageSize # 0 00
lineCount ← lineCount + 1;
Stream.PutChar [statsFile, Ascii.CR];
ENDLOOP; -- Complete Page

END; -- WriteFiles

```

```

-- Write the export information out to the second page
WriteExportInfo: PROCEDURE [aIP: LONG POINTER, aSP:
Parser.AtomStackPtr,

```

```

fSP: Parser.FileStackPtr, MaxFiles:
CARDINAL] =

```

```

BEGIN

```

```

-- Local variable declarations
currentCall: CARDINAL ← 0;
lastCall: CARDINAL ← 0;
tempIndex: CARDINAL ← 1;
tempFileCount: CARDINAL ← 0;
OK: BOOLEAN ← FALSE;
procCall: CARDINAL ← 237;
positionIndex: CARDINAL ← 0;
exportStackCount: CARDINAL ← 0;

exportStackPtr ← z.NEW[ExportStack];
-- Procedure starts
exportStackPtr ← z.NEW[ExportStack];
-- Collect missing information and write it out
WHILE tempIndex < aIP↑ 00

tempIndex ← tempIndex + 1;

IF aSP[tempIndex].idMark = procCall THEN
BEGIN
IF NOT String.Equal[aSP[tempIndex].IEatom,"-"] THEN
{
-- Only process items with a valid import entry
tempFileCount ← 0;
WHILE tempFileCount < MaxFiles 00

tempFileCount ← tempFileCount + 1;
IF fSP[tempFileCount].fileCount = 0 THEN EXIT;

-- Check if there is a matching export
IF String.Equal[aSP[tempIndex].IEatom,
fSP[tempFileCount].exportName]
THEN BEGIN
numString ←
String.CopyToNewString[aSP[tempIndex].inputString,z
];
[OK] ← CheckProcExists[aSP, fSP, tempFileCount,
numString];
String.FreeString[z,numString];
IF OK THEN EXIT; -- No action export information
available
END;

ENDLOOP; -- tempFileCount

IF NOT OK THEN
{
OK ← FALSE; -- Reset
exportStackCount ← exportStackCount + 1;
exportStackPtr[exportStackCount].missingProcLocation ←
aSP[tempIndex].Index;
loopCount ← exportStackCount;

WHILE loopCount > 1 00
loopCount ← loopCount - 1;
currentCall ←
exportStackPtr[loopCount].missingProcLocation;
lastCall ←
exportStackPtr[exportStackCount].missingProcLocation;
IF String.Equal[aSP[lastCall].inputString,
aSP[currentCall].inputString] THEN

```

```

{
  exportStackCount ← exportStackCount - 1;
  EXIT;
};

ENOLoop; -- loopCount
};
END;

ENOLoop; -- tempIndex

OutputExportInfo[exportStackCount, exportStackPtr, fSP, aSP];

z.FREE[exportStackPtr];

END; -- WriteExportInfo

-- OutputExportInfo: This procedure writes the information in the
-- exportStack out the data file
OutputExportInfo: PROCEDURE[ESC: CARDINAL, eSP:LONG POINTER TO
ExportStack,
fSP:Parser.FileStackPtr, aSP:Parser.AtomStackPtr]
=
BEGIN
  maxExportNameLength: CARDINAL ← 25;
  maxFileNameLength: CARDINAL ← 25;
  tempIndex: CARDINAL ← 0;
  loopCount: CARDINAL ← 0;
  titleSize: CARDINAL ← 7;

  WriteExportTitles[
lineCount ← lineCount + titleSize;

  WHILE loopCount < ESC 00

    loopCount ← loopCount + 1;
    lineCount ← lineCount + 1;
    IF lineCount MOD pageSize = 0 THEN
      {
        WriteExportTitles[
lineCount ← lineCount + titleSize;
      ];

      -- Write out line index number
      numString ←
String.ToNewString[Utils.CardToString[loopCount],z];
      CheckLength[maxIndexLength];
      StatsWrite[numString];
      String.FreeString[z, numString];
      StatsWrite[""];

      -- Write out file name where problem occurred
      StatsWrite[""];
      numString ←
String.ToNewString[fSP[aSP[tempIndex].fileNumber].file
Name,z];
      CheckLength[maxFileNameLength];
      StatsWrite[numString];
      String.FreeString[z, numString];

      -- Write out name of procedure called
      StatsWrite[""];
      numString ←
String.ToNewString[aSP[tempIndex].inputString,z];
      CheckLength[maxFileNameLength];
      StatsWrite[numString];
      String.FreeString[z, numString];

      Stream.PutChar[statsFile, Ascii.CR];

ENOLoop; -- loopCount;

-- Complete page
WHILE lineCount MOD pageSize # 0 00
  lineCount ← lineCount + 1;
  Stream.PutChar[statsFile, Ascii.CR];
ENDLoop; -- Complete Page

ENO; -- OutputExportInfo

-- Writes the title information at the top of each export info page
-- Write top of page title
WriteExportTitles: PROCEDURE =
BEGIN
  StatsWrite ["STRUCTURE CHART - TEXTURAL OUTPUT TABLE - MISSING
EXPORT INFORMATION"];
  Stream.PutChar[statsFile, Ascii.CR];
  StatsWrite
["=====
===="];
  Stream.PutChar[statsFile, Ascii.CR];
  StatsWrite ["The following is the list of unknown export
information. The files"];
  Stream.PutChar[statsFile, Ascii.CR];
  StatsWrite ["loaded in this session do not contain the exported
procedures that follow:"];

```

```

Stream.PutChar [statsFile, Ascii.CR];

-- Write identifier titles for export information fields
StatsWrite ["-----"-----"L"];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["LINE # MISSING EXPDRT      CALL MADE FROM
FILE      PROCEDURE CALLED "L"];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["-----"-----"L"];

Stream.PutChar [statsFile, Ascii.CR];
END; -- WriteExportTitles

-- Writes the title information at the top of each Structure
ChartInfo page
-- Write top of page title
WriteSCTitles: PROCEDURE =
BEGIN
  titleSize: CARDINAL ← 5;

  -- Write identifier titles for export information fields
  Stream.PutChar [statsFile, Ascii.CR];
  StatsWrite ["STRUCTURE CHART - SEMI GRAPHICAL OUTPUT TABLE
Indented Procedure Call Levels" L];
  Stream.PutChar [statsFile, Ascii.CR];
  StatsWrite ["-----"-----"
-----" L];
  Stream.PutChar [statsFile, Ascii.CR];
  StatsWrite ["Line Called From Filename  Interface Name  Declared
in Filename  Lvl  SLC  1  2  3  4  5  6  7  8
9  10  11  12 " L];
  Stream.PutChar [statsFile, Ascii.CR];
  StatsWrite ["-----"-----"
-----" L];
  Stream.PutChar [statsFile, Ascii.CR];
  lineCount ← lineCount + titleSize
END; -- WriteSCTitles

-- Writes a string to the statistics output file
StatsWrite: PROCEDURE [str: LDNG STRING] =
BEGIN
  Stream.PutString [statsFile, str];
END; -- StatsWrite

-- Checks the length of the string and adds in spaces if short
CheckLength: PROCEDURE [length: CARDINAL] =
BEGIN
  tempIndex: CARDINAL ← 0;

```

```

-- Local variable
size: CARDINAL;

-- Procedure body
size ← String.Length[numString];
IF size < length THEN AddSpaces[size,length];
IF size > length THEN ReduceLength[size,length];
END; -- CheckLength;

AddSpaces: PROCEDURE[size: CARDINAL, length: CARDINAL] =
BEGIN
  size ← length - size; -- Determine how many spaces need to be
  added
  WHILE size > 0 DO
    String.AppendStringAndGrow[@numString, ,z];
    size ← size - 1;
  ENDDO;

END; -- AddSpaces

ReduceLength: PROCEDURE[size: CARDINAL, length: CARDINAL] =
BEGIN
  -- Substring allows the extraction of a portion of a string
  subStringDescr: String.SubStringDescriptor;
  tempString: LDNG STRING ← NIL;

  -- Procedure body
  subStringDescr ← [numString,0,length];
  tempString ← String.MakeString[z,length];
  String.AppendSubString[tempString,@subStringDescr];
  String.FreeString [z, numString];
  numString ← String.CopyToNewString[tempString,z];
  String.FreeString [z,tempString];

END; -- ReduceLength

```

```

-- Check if the procedure are declared in the exporting program
CheckProcExists: PROCEDURE[aSP: Parser.AtomStackPtr,
fSP: Parser.FileStackPtr,
tempFileCount: CARDINAL,
numString: LDNG STRING
RETURNS[BDDLEAN] =
BEGIN
  -- Local Variables
  tempIndex: CARDINAL ← 0;

```

```

proc:
CARDINAL + 37;
foundProcDeclaration: BOOLEAN + FALSE;

--Procedure begins
tempIndex + fSP[tempFileCount].startIndex;

WHILE tempIndex < fSP[tempFileCount].endIndex DO
IF aSP[tempIndex].idMark = proc THEN
BEGIN
IF String.Equal[numString,aSP[tempIndex].inputString] THEN
{
foundProcDeclaration + TRUE;
aSP[posIndex].declLocation + tempIndex;
posIndex + tempIndex;
EXIT};
END;

tempIndex + tempIndex + 1;
ENDLOOP; -- tempIndex

RETURN [ foundProcDeclaration];

END; -- Procedure CheckProcExists

-- TabularOutput: This procedure controls the output of the
-- final Structure Chart table in a tabular output format.
-- Clients, subordinates and Structure Chart levels are
-- all detailed in StructChart.data
TabularOutput: PROCEDURE [
aSP:Parser.AtomStackPtr,fSP:Parser.FileStackPtr,
debugg:BOOLEAN, repeats:BOOLEAN] =
BEGIN
-- Local variables
tempStackCount: CARDINAL + 0;
currentLevel: CARDINAL + 1;
currentCall: CARDINAL + 1;
lastCall: CARDINAL + 1;
location: CARDINAL + 0;
found: BOOLEAN + FALSE;
SLC: BOOLEAN + FALSE;
titleLabel: CARDINAL + 8;
loopBackCount: CARDINAL + 0;

-- Procedure start
WriteTableTitles[];
lineCount + lineCount + titleLabel;

-- First look for straight line code
WHILE tempStackCount < fullCount DO
IF fullStackPtr[tempStackCount].SLC THEN
{
SLC + TRUE;
OutputTableLine[tempStackCount,fSP,aSP,SLC,titleSize];
}
ENDLOOP; -- tempStackCount

ENDLOOP;

```

```

currentLevel ← currentLevel + 1;
ENDLOOP; -- currentLevel

-- Complete page
WHILE lineCount MOD pageSize = 0 DO
    lineCount ← lineCount + 1;
    Stream.PutChar [statsFile, Ascii.CR];
ENDLOOP; -- Complete page

END; -- TabularOutput

-- OutputTableLine: This procedure outputs a line for the tabular
-- representation of the structure chart.
OutputTableLine: PROCEDURE [tempStackCount:
    CARDINAL, fsp: Parser.FileStackPtr,
    asp: Parser.AtomStackPtr, SLC: BOOLEAN,
    titleSize: CARDINAL] =
BEGIN
    callLocation: CARDINAL ← 0;
    fileNumber: CARDINAL ← 0;
    size: CARDINAL ← 0;
    tSC: CARDINAL ← tempStackCount;
    level: CARDINAL ← fullStackPtr[tSC].callLevel;
    firstLine: BOOLEAN ← FALSE;

    -- WriteRepeatInfo: This local procedure outputs
    -- the repeat information.
    WriteRepeatInfo: PROCEDURE =
    BEGIN
        numString ← String.CopyToNewString["", z];
        IF fullStackPtr[tSC].firstOfTwin OR
        fullStackPtr[tSC].twinRepeat THEN
            String.AppendStringAndGrow[@numString, "T", z];
        IF fullStackPtr[tSC].firstOfLevel OR
        fullStackPtr[tSC].levelRepeat THEN
            String.AppendStringAndGrow[@numString, "L", z];
        IF fullStackPtr[tSC].firstOfGlobal OR
        fullStackPtr[tSC].globalRepeat THEN
            String.AppendStringAndGrow[@numString, "G", z];
        CheckLength[maxRepeatLength];
        StatsWrite [numString];
        String.FreeString [z, numString];
        StatsWrite[" "];
    END; -- WriteRepeatInfo

    -- Write out SLC status
    IF fullStackPtr[tSC].SLC THEN StatsWrite["SLC "]
    ELSE
        {
            -- Write out procedure level
            numString ←
                String.CopyToNewString[Utils.CardToString[level], z];
            CheckLength[maxLevelLength];
            StatsWrite [numString];
            String.FreeString [z, numString];
        };
    };

    -- Write out the repeat status
    WriteRepeatInfo[];

    -- Write out client procedure name
    callLocation ← fullStackPtr[tSC].callLocation;
    numString ←
        String.CopyToNewString[asp[callLocation].inputString, z];
    CheckLength[maxProcLength];
    StatsWrite [numString];
    String.FreeString [z, numString];
    StatsWrite [" "];

    -- Write out Importing interface name
    numString ← String.CopyToNewString[asp[callLocation].IEatom, z];
    CheckLength[maxIEidentLength];
    StatsWrite [numString];
    String.FreeString [z, numString];
    StatsWrite [" "];

    -- Write file name that procedure resides in
    fileNumber ← FindFileNumber[fSP, callLocation];
    numString ← String.CopyToNewString[fSP[fileNumber].fileName, z];
    size ← String.Length[numString];
    ReduceLength[size, size - 5];
    CheckLength[maxFileLength - 5];
    StatsWrite [numString];
    String.FreeString [z, numString];

    -- Now find the subordinates
    tSC ← tSC + 1;

    WHILE fullStackPtr[tSC].callLevel > (level + 1) DO
        IF fullStackPtr[tSC].callLevel = (level + 1) AND
        NOT fullStackPtr[tSC].twinRepeat THEN
            {
                IF NOT firstLine THEN
                    {
                        lineCount ← lineCount + 1;
                        Stream.PutChar [statsFile, Ascii.CR];
                        IF lineCount MOD pageSize = 0 THEN
                            {
                                WriteTableTitles[];
                                lineCount ← lineCount + titleSize;
                            };
                        numString ← String.CopyToNewString[" ", z];
                        CheckLength[maxSubMargin];
                        StatsWrite [numString];
                        String.FreeString [z, numString];
                    };
                -- Write out procedure level
            }
        }
    }

```



```

numString ←
String.CopyToString[Utils.CardToString[level+1]
.z];
StatsWrite [numString];
CheckLength[maxLevelLength];
String.FreeString [z, numString];
StatsWrite [" " "L"];

-- Write out repeat status
WriteRepeatInfo[];

-- Write out subordinate procedure name
callLocation ← fullStackPtr[tSC].callLocation;
numString ←
String.CopyToString[aSP[callLocation].inputStri
ng,z];
CheckLength[maxProcLength];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" " "L"];

-- Write out Importing interface name
numString ←
String.CopyToString[aSP[callLocation].IEatom,z]
;
CheckLength[maxIEidentLength];
StatsWrite [numString];
String.FreeString [z, numString];
StatsWrite [" " "L"];

-- Write file name that procedure resides in
fileNumber ← FindFileNumber[fSP,callLocation];
numString ←
String.CopyToString[fSP[fileNumber].fileName,z]
;
size ← String.Length[numString];
ReduceLength[size, size - 5];
CheckLength[maxFileLength - 5];
StatsWrite [numString];
String.FreeString [z, numString];

firstLine ← FALSE;
};

tSC ← tSC + 1;

ENDLOOP; -- fullStackPtr

Stream.PutChar [statsFile, Ascii.CR];
lineCount ← lineCount + 1;
IF lineCount MOD pageSize = 0 THEN
{
WriteTableTitles[];
lineCount ← lineCount + titleSize;
};

END; -- OutputTableLine

```

```

-- WriteTableTitles: This procedure writes out the titles for
-- the structure chart tabular format.
WriteTableTitles: PROCEDURE =
BEGIN
Stream.PutChar [statsFile, Ascii.CR];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["STRUCTURE CHART - TABULAR OUTPUT LISTING" "L"];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["===== " "L"];
Stream.PutChar [statsFile, Ascii.CR];

StatsWrite [" CLIENT SUBORDINATES" "L"];

Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["-----
-----" "L"];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["LVL REP PROCEDURE NAME INTERFACE NAME FILE
NAME FILE NAME " "L"];
Stream.PutChar [statsFile, Ascii.CR];
StatsWrite ["-----
-----" "L"];
Stream.PutChar [statsFile, Ascii.CR];
END; -- WriteTableTitles

END. -- StructChartImpl

```

```

-- File: UtilsImpl.mesa - last edit:
-- Bonikowski:HenrB01C:Xerox 24-Apr-89 22:06:50
--
*****
-- File: UtilsImpl.mesa
--
-- Copyright (C) 1989 by Xerox Corporation. All rights reserved.
--
-- David Egerton:Wbst129
--
-- This module contains the programs for several utility routines and
-- data
-- structures used in the DesignExtractor. Information from John
-- Lewis
-- program of 1987 was adopted for use here.
--
-- Copyright (C) 1987 to 1992 by Xerox Corporation. All rights
-- reserved.
--
*****
OIRECTORY

Real USING [Fix, FixC],
String USING [Compare, MakeString],
Utils USING [tempZone];

UtilsImpl: PROGRAM
    IMPORTS Real, String, Utils
    EXPORTS Utils =
    BEGIN

StringEqual: PUBLIC PROCEDURE [str1, str2: LONG STRING] RETURNS
[BOOLEAN] =
    -- Returns a boolean indicating if str1 has the equivalent string
    value
    as str2
    BEGIN
        comp: INTEGER ← String.Compare[str1, str2, FALSE];
        RETURN [(comp = 0)];
        ENO; -- StringEqual

StringLess: PUBLIC PROCEDURE [str1, str2: LONG STRING] RETURNS
[BOOLEAN] =
    -- Returns a boolean indicating if str1 is "less than" str2 in a
    comparison of ASCII values
    BEGIN
        comp: INTEGER ← String.Compare[str1, str2, FALSE];
        RETURN [(comp = -1)];
        ENO; -- StringLess

CardToString: PUBLIC PROCEDURE [num: CARDINAL] RETURNS [LONG
STRING] =
    -- Converts a cardinal value into a string value.
    BEGIN
        numstring: LONG STRING ← String.MakeString [Utils.tempZone, 10];
        scan, whole: LONG INTEGER;
        temp: INTEGER;
        strIndex: CARDINAL;
        digitchar: CHARACTER;
        stringEmpty: BOOLEAN ← TRUE;
        strIndex ← 0;
        IF ( num = 0 ) THEN
            BEGIN
                numstring [0] ← '0';
                numstring.length ← 1;
            ENO
        ELSE BEGIN
            whole ← num;
            FOR scan ← 1000000, scan / 10 UNTIL (scan < 1) DO
                temp ← Real.FixC [ whole / scan ];
                digitchar ← VAL [temp + 48];
                IF (NOT (digitchar = '0 AND stringEmpty)) THEN
                    BEGIN
                        numstring [strIndex] ← digitchar;
                        numstring.length ← numstring.length + 1;
                        strIndex ← strIndex + 1;
                        stringEmpty ← FALSE;
                    ENO
                ELSE BEGIN
                    IF ( scan <= 100 ) THEN
                        BEGIN
                            numstring [strIndex] ← ;
                            numstring.length ← numstring.length + 1;
                            strIndex ← strIndex + 1;
                        ENO;
                    whole ← whole MOD scan;
                ENOLOOP;
            END
        END
    END

```

```

END;

RETURN [numstring];

END; -- CardToString

RealToString: PUBLIC PROCEDURE [num: REAL] RETURNS [LONG STRING] =
-- Converts a real value into a string. The fractional part is
-- truncated to three digits.
BEGIN
    numstring: LONG STRING ← String.MakeString [Utils.tempZone,
20];
    strIndex: CARDINAL;
    digitchar: CHARACTER;
    scan, whole: LONG INTEGER;
    fract: REAL;
    temp: INTEGER;
    stringEmpty: BOOLEAN ← TRUE;

    -- whole part
    whole ← Real.Fix [num];
    fract ← (num - whole) * 10;
    strIndex ← 0;

    FOR scan ← 10000000, scan / 10 UNTIL (scan = 0) DO
        temp ← Real.FixC [ whole / scan ];
        digitchar ← VAL [temp + 48];
        IF (NOT (digitchar = '0 AND stringEmpty))
            THEN BEGIN
                numstring [strIndex] ← digitchar;
                numstring.length ← numstring.length + 1;
                strIndex ← strIndex + 1;
                stringEmpty ← FALSE;
            END
        ELSE BEGIN
            IF ( scan <= 1000000 ) THEN
                BEGIN
                    numstring [strIndex] ← ' ';
                    numstring.length ← numstring.length + 1;
                    strIndex ← strIndex + 1;
                END;
            END;
            whole ← whole MOD scan;
        ENDLOOP;

        -- fractional part
        numstring [strIndex] ← '.';
        numstring.length ← numstring.length + 1;
        strIndex ← strIndex + 1;

        FOR scan IN [1..3] DO
            digitchar ← VAL [Real.FixC [fract] + 48];
            numstring [strIndex] ← digitchar;
            numstring.length ← numstring.length + 1;
            strIndex ← strIndex + 1;
            fract ← (fract - Real.FixC [fract]) * 10;
        ENDLOOP;

        RETURN [numstring];
    END; -- RealToString

END. -- PROGRAM UtilsImpl

```

This appendix details the PGS (Parser Grammar Specification) for the Pass1 parser. The Grammar is essentially the same as that used in the Mesa compiler specification. However further fields have been added where the Mesa Compiler does not cover the function. In Pass1 the Atom procedure compares each of these keywords and assigns a number to atomStack.symbolNumber if there is a match.

#	ID symbol	#	ID symbol	#	ID symbol	#	ID symbol	#	ID symbol	#	ID symbol
1	Identifier	31	RECORD	61		91		121		151	
2	#	32	POINTER	62	MACHINE	92		122	FREE	152	
3		33	VAR	63	DEPENDENT	93		123		153	
4		34	ARRAY	64	DIRECTORY	94		124		154	
5	"String"	35	SEQUENCE	65	IMPORTS	95		125		155	
6	"lString"	36		66	EXPORTS	96		126		156	
7		37	PROCEDURE	67		97		127		157	
8		38		68		98		128		158	
9	'	39		69	USING	99		129		159	
10	;	40		70	PUBLIC	100		130		160	
11	:	41		71		101	NIL	131		161	
12		42		72	ENTRY	102		132		162	
13		43	PROGRAM	73		103		133			
14	<-	44	MONITOR	74		104	IF	134	OPEN		
15	=	45	DEFINITION	75		105	THEN	135		200	parameter
16		46	ZONE	76		106		136		201	other
17		47		77		107		137		202	BOOLEAN
18		48	LONG	78		108		138)	203	CARDINAL
19		49	TYPE	79		109		139]	204	TRUE
20		50		80		110		140	}	205	FALSE
21		51	TO	81		111		141	END	206	NAT
22		52		82		112		142		207	CONFIG
23		53	UNCOUNTED	83		113		143		208	LONG STRING
24		54		84		114		144	(
25		55		85		115		145	[237	procCall
26	^	56		86		116		146	{	238	paramCall
27	.	57	RETURNS	87		117	RETURN	147	BEGIN		
28	@	58		88		118	EXIT	148	DO		
29		59	NEW	89		119		149		500	variable
30		60		90		120		150		501	variableDecl

Table C.1 - ID number versus ID symbol

Notes: 1. Capitalized ID symbols are MESA KEYWORDS 2. Lower case ID symbols have meaning to DesignExtractor
3. Other symbols are MESA operators

APPENDIX D - VARIABLE TYPES

The variables in the DesignExtractor program are given a designation to help distinguish between their relative functions. The variable types are held as a field of DFDDStack declared in the DataFlowImpl. The usage of the field can be seen in the output listing for DFDDTable which is described in Sections 2.1.3.3.1 & 2.1.3.3.2. The field is a special defined type, the designators below are used in the field:

CallParam: A variable designated as a *Call Parameter* is a variable that is in the list of parameters associated with a procedure call.

DataBase: A variable designated as a *Database* is one that is none of the other designations.

DataFlow: A variable designated as a *DataFlow* is the same as DataBase, this designation will never be seen as Database gets allocated first.

Dummy: The default value for a variable prior to resolution is *Dummy*.

External: A variable designated as *External* is one that is known to be in the external list specified in Pass2 or ProgramParserImpl. These variable would be seen at the context level as final destinations or sources of data to the program.

FormalParam: A variable designated *Formal Parameter* is one that is in the declarations list of a procedure.

IntCallParam: A variable designated as an *Internal Call Parameter* is one that is used as a parameter of a call that resides inside a procedure body.

ReturnValue: A variable designated as *Return Value* is the variable that is actually returned when a procedure is exited.

FormalReturn: A variable designated as *Formal Return* is one that is in the declarations list of a procedure header, under the return section.

APPENDIX E - VARIABLE SCOPES

The variables in the DesignExtractor program are given a designation to help distinguish the various scope options. The variable scopes are held as a field of atomStack the main program reference database. The usage of the field can be seen in the output listing for DFDDTable which is described in Sections 2.1.3.3.1 & 2.1.3.3.2. The field is a character type, designators below are used in the field:

D: Dereference. The scope of a dereferenced variable is tied up with its parent. (eg aSP[index].varScope). varScope is a dereferenced variable whose parent is aSP. Therefore only the parents need be studied for scope, the D reference can be bypassed.

E: External. An External database would be seen at the context level of the implementation. Examples of external variables or databases are file outputs.

G: Global. Global variables are those that can be accessed from any procedure within a particular impl. They are declared at the header of the impl file before any executable code.

I: Instantiated. These are the variables declared in the procedure headers. Procedure calls will substitute the header naming convention for the call naming convention.

L: Local. Confined to the current procedure.

M: Imported. The parser got to the start of the file and no variable declaration match was found. This variable must therefore be imported.

P: Public. The variable is declared as public.

R: Return parameter. This variable is the one declared in the procedure header as a return parameter.

APPENDIX F - REPEATS

During the first implementation of the Structure Chart graphical output (Section 2.1.3.2.1) it was discovered that the output was a little difficult to read because of the continuous repetition of procedure calls. From a structure chart perspective only the first call from a parent is relevant. Therefore a program module was written to suppress these items. The user interface has the option to display or suppress. The default is suppress repeats (see Section 2.1.1.3). The "StudyRepeats" procedure in Structure chart performs the analysis.

There were three kinds of repeat types that were discovered and suppressed, as follows:

Twin repeat: This occurs when a procedure call is made and then immediately followed by a second call to the same procedure.

Level repeat: The program registers the first occurrence of a procedure at a particular Structure Chart level, then each time it is repeated at that level the subsequent occurrences are suppressed. Note if a call is made at a lower level then the first occurrence at that level is marked for output and then suppressed etc.

Global repeat: On a global level procedure calls are registered for their first occurrence and subsequent occurrences. However suppression does not occur for these.

APPENDIX G - LIST TYPES

There are various list types that are studied during the parsing algorithms. In ProgramParserImpl Pass 2, where contexts are identified all the tokens between a begin and end bracket are allocated a number appropriate to the list type. This enables later, during Pass3 where identifiers are resolved, for a procedure called "CheckListMark" to understand the context of a variable. Assignments of list type numbers are allocated as each variable is studied and placed into the "idMark" field of the atomStack.

The parser finds the appropriate token placed just before the list starts, usually identified by open bracket [, then uses this to determine the appropriate numerical designator for idMark. The following allocations are made:

List Type	identifying token	Allocation to idMark	Comments
parameter	37 = PROCEDURE 38 = PROC	parameter = 200	List mark for a formal procedure declaration parameter list.
returns	57 = RETURNS	returns = 57	List mark for a formal procedure returns parameter list
using	69 = USING	using = 69	List mark for the whole of the using clause section declared just before the imports/export list at the top of each program file.
return	117 = return	return = 117	List mark for the series of data items in the list following the return command at the close of a procedure body.
other	anything else	other = 201	This list mark is allocated to all other undefined lists. This helps identify that there is a list present but that it is not resolved at this time
directory	64 = directory	directory = 64	The list mark is given the designator for the directory list that is present at the top of each program modules.
imports	65 = imports	imports = 64	The list mark for the imports list
exports	66 = exports	exports = 66	The list mark for the exports list

Table G.1 - Mapping of list marks to idMark field