(NASA-TM-84726) THE FORTRAN STATIC SOURCE CODE ANALYZER PROGRAM (SAP) USER'S GUIDE, REVISION 1 (NASA) 134 p HC A07/MF A01
CSCL 09B
N83-19495
Unclas
G3/61 02146

SOFTWARE ENGINEERING LABORATORY SERIES SEL-78-102

FORTRAN STATIC SOURCE CODE ANALYZER PROGRAM (SAP) USER'S GUIDE (REVISION 1)

SEPTEMBER 1982

NASA
National Aeronautics and
Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

FOREWORD

The Software Engineering Laboratory (SEL) is an organization sponsored by the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC) and created for the purpose of investigating the effectiveness of software engineering technologies when applied to the development of applications software. The SEL was created in 1977 and has three primary organization members:

NASA/GSFC (Systems Development and Analysis Branch)
The University of Maryland (Computer Sciences Department)
Computer Sciences Corporation (Flight Systems Operation)

The goals of the SEL are (1) to understand the software development process in the GSFC environment; (2) to measure the effect of various methodologies, tools, and models on this process; and (3) to identify and then to apply successful development practices. The activities, findings, and recommendations of the SEL are recorded in the Software Engineering Laboratory Series, a continuing series of reports that includes this document. A version of this document was also issued as Computer Sciences Corporation document CSC/SD-82/6044.

Contributors to this document include

William Decker	(Computer Sciences Corporation)
Wayne Taylor	(Computer Sciences Corporation)

Other contributors include

Suellen Eslinger	(Computer Sciences Corporation)
Frank McGarry	(Goddard Space Flight Center)
Phil Merwarth	(Goddard Space Flight Center)

Single copies of this document can be obtained by writing to

Frank E. McGarry
Code 582.1
NASA/GSFC
Greenbelt, Maryland 20771

PRECEDING PAGE BLANK NOT FILMED

ABSTRACT

This document presents the FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 1). SAP is a software tool designed to assist Software Engineering Laboratory (SEL) personnel in conducting studies of FORTRAN programs. SAP scans FORTRAN source code and produces reports that present statistics and measures of statements and structures that make up a module. This document is a revision of the previous SAP user's guide, Computer Sciences Corporation document CSC/TM-78/6045. SAP Revision 1 is the result of program modifications to provide several new reports, additional complexity analysis, and recognition of all statements described in the FORTRAN 77 standard. This document provides instructions for operating SAP and contains information useful in interpreting SAP output.

PRECEDING PAGE BLANK NOT FILMED

ORIGINAL PAGE IS
OF POOR QUALITY

TABLE OF CONTENTS

PRECEDING PAGE BLANK NOT FILMED

ORIGINAL PAGE IS
OF POOR QUALITY

TABLE OF CONTENTS (Cont'd)

Appendix A - Cont'd)

ORIGINAL PAGE IS
OF POOR QUALITY

TABLE OF CONTENTS (Cont'd)

ORIGINAL PAGE IS
OF POOR QUALITY

LIST OF ILLUSTRATIONS

LIST OF TABLES

ORIGINAL PAGE IS
OF POOR QUALITY

LIST OF TABLES (Cont'd)

ORIGINAL PAGE IS
OF POOR QUALITY

SECTION 1 - INTRODUCTION

The FORTRAN Static Source Code Analyzer Program (SAP) automatically produces statistics on occurrences of statements and structures within FORTRAN program modules and provides a facility for reporting these statistics. SAP is available in versions to run on either a PDP-11/70 or a VAX-11/780 computer. This document is a revision of the previous SAP user's guide, Computer Sciences Corporation (CSC) document CSC/TM-78/6045, which describes SAP Version 1. SAP Version 2 is a result of program modifications to provide several new reports, additional complexity analysis, and recognition of all statements described in the American National Standards Institute Programming Language FORTRAN standard (FORTRAN 77), ANSI X3.9-1978 (Reference 1).

SAP accepts as input syntactically correct FORTRAN source code written in the FORTRAN 77 standard language. In addition, code written using features in the following languages is also accepted: PDP-11 FORTRAN IV or FORTRAN IV-PLUS (References 2 and 3); VAX-11 FORTRAN (References 4 and 5); IBM S/360 FORTRAN IV Level H Extended, with the exception of the S/360 FORTRAN DEBUG Facility statements (References 6 and 7); and Structured FORTRAN (Reference 8).

SAP operates in an interactive environment in which the user is prompted for a file name to specify the source code to be analyzed. The file name may be modified by optional control switches, which are used to specify the types of processing to be performed and the specific output files to be created.

The program uses two external permanent files: a keywords file and a statistical weights file. The keywords file provides flexibility in classifying statements as executable or nonexecutable. The statistical weights file is used in the computation of the Software Engineering Laboratory (SEL)

ORIGINAL PAGE IS
OF POOR QUALITY

complexity (Section 3.4). The user may specify a different set of weights and thus redefine the composition of this measure of complexity.

SAP can produce three types of output: reports formatted for the line printer, files containing statistical results in a format readable by other analysis programs, and error and warning messages directed to the user's terminal.

This document is a detailed user's guide for SAP. Section 2 presents the instructions for operating the SAP program. The information presented describes starting the program, controlling the source code processing, stopping the program, and examining SAP output. Section 3 presents an overview of the SAP system, describing the content of SAP output files and presenting a simplified description of how source code is processed.

Four appendixes are provided to describe detailed information for researchers interested in how SAP collects and calculates statistics. Appendix A descibes how each FORTRAN statement type is processed. Appendix B describes how Halstead's measures (Reference 9) are gathered. Appendix C descibes how McCabe's measure (Reference 10) and the number of decisions are counted. Appendix D describes how the statistical weights file and the user complexity "stubs" (UCPLX1 and UCPLX2) are used to calculate the user's own definition of complexity for a module.

An additional appendix, Appendix E, presents the error messages produced by SAP, along with an explanation of the probable cause of each error.

ORIGINAL PAGE IS
OF POOR QUALITY

SECTION 2 - SAP PROGRAM OPERATION

The SAP program is an interactive program and is operational on the PDP-11/70 and the VAX-11/780. This section describes the use of SAP on either machine. Only in cases in which a difference exists in operating procedure will two examples (one for the VAX-11/780 and one for the PDP-11/70) be given. This section is divided into three parts: a description of SAP input files and how to prepare them for processing, a description of the user's interactive control of SAP, and a description of SAP output files.

2.1 SAP INPUT FILES

The input to SAP is a Files-11 American Standard Code for Information Interchange (ASCII) file containing syntactically correct FORTRAN source code. SAP accepts code written in the FORTRAN 77 standard language. In addition, code written using the features in the following languages is also accepted: PDP-11 FORTRAN IV or FORTRAN IV-PLUS (References 2 and 3); VAX-11 FORTRAN (References 4 and 5); IBM S/360 FORTRAN IV Level H Extended, with the exception of the S/360 FORTRAN DEBUG Facility statements (References 6 and 7); and structured FORTRAN (Reference 8).

Source code brought from non-DEC computers should be on tape in a fixed-block, fixed-record-length format. The International Business Machines Corporation (IBM) utility IEBGENER (Reference 11) or a suitable update program (for example, PACKUPD or PANVALET) may be used to create sequential tapes in this format.

2.1.1 READING FOREIGN TAPES ON THE PDP-11/70

On the PDP-11/70, the local utility, TRN, is used to move and translate a foreign tape file to a Files-11 formatted disk file. TRN assumes that each physical record on the tape is composed of multiple 80-byte card images. Trailing

ORIGINAL PAGE IS
OF POOR QUALITY

blanks are removed from each card image before it is written to disk. To run TRN, the following commands are entered (user input is underlined and <CR> indicates a carriage return):

```
>ALL MM0: <CR>
>DENS <CR>
PLEASE VERIFY THAT TAPE IS LOADED ON DRIVE
THAT YOU HAVE ALLOCATED BEFORE PROCEEDING.
ENTER TAPE UNIT # (0 OR 1)> MM0 <CR>
ENTER TAPE DENSITY (800 OR 1600) > 1600 <CR>
TAPE DRIVE MM0:  HAS BEEN SET TO A DENSITY OF 1600 BPI.
>TRN Outfile/FI:n/EB <CR>
```

where Outfile is the destination file specifier and n is the relative file number on the tape. The /EB switch is optional and is used only if the tape file contains Extended Binary Coded Decimal Interchange Code (EBCDIC) code.

2.1.2 READING FOREIGN TAPES ON THE VAX-11/780

On the VAX-11/780, the local utility, TAPECOPY, is used to move and translate foreign tape files to Files-11 formatted disk files. TAPECOPY assumes that each physical record on the tape is composed of multiple 80-byte card images. Trailing blanks are removed from each card image before being written to disk. To run TAPECOPY, enter the following commands (user input is underlined and <CR> indicates a carriage return):

```
$ ALLOCATE MTA0: <CR>
$ MOUNT/FOREIGN/DENSITY=1600 MTA0: - <CR>
$_ LABEL LOGNAM <CR>
$ RUN DBB1:[FDYN]TAPECOPY <CR>
```

TAPECOPY will prompt the user for the tape file number, EBCDIC-to-ASCII translation, and destination file name.

ORIGINAL PAGE IS
OF POOR QUALITY

2.2 SAP EXECUTION

SAP is executed interactively on the PDP-11/70 and VAX-11/780 as follows:

```
(PDP)    > RUN DB1:[213,2]SAP
(VAX)    $ RUN DBB1:[TOOLS]SAP
```

SAP execution is divided into two sequential stages: source code file analysis and project analysis. The following sections discuss the operation of SAP in each phase. The instructions presented in the remainder of this section are independent of the particular machine on which SAP is running.

2.2.1 SAP SOURCE CODE FILE ANALYSIS STAGE

After the user has entered the appropriate RUN command shown above, the program will respond with the following prompt:

```
SAP>
```

The user specifies and controls SAP processing in this stage by entering the names of files to be analyzed and appending control switches as required. To exit the SAP source code file analysis stage, the user enters a control Z (^Z) in response to the SAP> prompt. SAP will then proceed to the project analysis stage.

The following general format is used to specify a source code input file and processing options:

```
SAP> fileinl/S1/S2.../SN <CR>
SAP>
```

where fileinl is the source input file specifier and /S1 through /SN are control switches. The file specifier may include a device and directory name if the user wishes to refer to a device or directory that is not the current default. The control switches are used to control SAP listing output and to direct SAP to use some external files in its

ORIGINAL PAGE IS
OF POOR QUALITY

processing. Tables 2-1 and 2-2 list the legal control switches.

SAP control switches are used to specify output listing contents and the use of external files. The switches are turned on (/XX) and off (/-XX) by appending the switch to the end of a file specification. Once a switch setting has been specified, the switch remains at that setting until respecified or until the end of a run.

In the following sequence, the user specifies that file DOTEST.FOR is to be processed with the /MO and /GB switches on and all other switches set to off (the default), file IFTEST.FOR is processed with no module statistics output, and file SHORTEST.FOR is to be processed with INCLUDE statements expanded and no module statistics output.

```
RUN SAP
SAP> DOTEST.FOR
SAP> IFTEST.FOR/-MO
SAP> SHORTEST.FOR/XP
  :
  :
```

The following should be noted when specifying SAP control switches:

- /MO Control Switch. When turned on, this switch specifies that the entire module statistics page is to be produced for each module regardless of the settings for the module paragraph switches (/EC, /CO, /SC, /ST, /CS, /AS, /SP, and /CA). When the /MO switch is turned off (/-MO), only the paragraphs with the corresponding switch set to on will appear on the module statistics page.

- /HL Control Switch. This switch controls the module operator/operand listing. This report always starts at the top of a separate page following the module statistics page if one is produced. This listing is not considered

ORIGINAL PAGE IS
OF POOR QUALITY

Table 2-1. SAP Listing Switches

Switch	Default	Description
/MO	/MO	Print entire module statistics summary
/EC	/-EC	Print module external communications paragraph (if /-MO in effect)
/CO	/-CO	Print module commenting paragraph (if /-MO in effect)
/SC	/-SC	Print module statement class counter paragraph (if /-MO in effect)
/ST	/-ST	Print module statement type counter paragraph (if /-MO in effect)
/CS	/-CS	Print module control statement breakdown paragraph (if /-MO in effect)
/AS	/-AS	Print module assignment statement breakdown paragraph (if /-MO in effect)
/SP	/-SP	Print module specification statement breakdown paragraph (if /-MO in effect)
/CA	/-CA	Print module complexity analysis paragraph (if /-MO in effect)
/HL	/-HL	Print module operator/operand summary paragraphs
/GB	/GB	Print global summary of statistics for input file
/LI	/-LI	Print source code listing
/DU	/-DU	Print formatted dump of the symbol taole

ORIGINAL PAGE IS
OF POOR QUALITY

Table 2-2. SAP External File Use Switches

Switch	Default	Description
/XP	/-XP	Expand (DEC) INCLUDE statements before processing
/UW	/-UW	Use an alternate file for SEL complexity weights
/DB	/-DB	Write module statistics to SAP data base file
/SL	/-SL	Write module statistics to SAP sequential output file, ALL.SAP

ORIGINAL PAGE IS
OF POOR QUALITY

part of the module summary and so is not affected by the setting of the /MO switch.

- /GB Control Switch. This switch controls the global summary statistics report. A module directory and a global statistics report will appear for each source input file while the /GB switch is on.

- /XP Control Switch. This switch instructs SAP to expand all INCLUDE statements (PDP and VAX FORTRAN only) encountered. INCLUDE statements will be expanded to a nesting depth of three. PANVALET ++INCLUDE statements cannot be expanded. Failure to expand INCLUDE statements may result in misleading statistical values.

This switch may be used to combine many source code files into one file. The user creates a small file consisting only of INCLUDE statements that refer to the files to be combined. When this file is processed by SAP with the /XP control switch set to on, SAP processes the files named on the INCLUDE statements as if they were one large sequential file. This procedure is a flexible and efficient way to repeatedly process the source code for a complete system. The advantages of this technique are that source code files can be added or removed easily by changing only one line of the file, the most recent version of the file is always used, and there is only one extra file to maintain.

- /UW Control Switch. This switch instructs SAP to use an alternate statistical weights file as the source of weights used in computing the SEL complexity (Section 3.4). After using the /UW switch, specifying /-UW will cause SAP to revert to the default statistical weights file.

- /DB Control Switch. This switch specifies that a SAP "data base" file is to be used to store statistical data. This file can receive data from the analysis of several source input files. When the user specifies this

ORIGINAL PAGE IS
OF POOR QUALITY

switch, SAP prompts the user for the name of a file to be used as the data base. If the file name does not refer to an existing data base file, SAP will prompt the user for the number of records to allocate for the file. Two records are used to store the statistics for each module, so that the user should specify at least twice the anticipated module count for the number of records.

After SAP has located or created the data base file, the user will be prompted for a project character. This character will identify the group of modules to be entered into the data base during this session with SAP.

If the user does not enter a character or enters a blank character, an asterisk will be used as the project character.

The data written into a data base file during the source code file analysis stage is available for the project analysis stage.

- /SL Control Switch. This switch instructs SAP to write statistical data to a sequential output file for possible use by other analysis programs. When this switch is specified, a file, ALL.SAP, is either created or opened for extension in the user's directory.

After locating or creating ALL.SAP, SAP will prompt the user for a project name up to eight characters in length, and a two-character subsystem prefix. These identifiers will be included on each record written to ALL.SAP during the session.

2.2.2 SAP PROJECT ANALYSIS STAGE

The second stage of SAP execution produces an analysis of module complexities from the data written to the SAP data base files during the source code file analysis stage.

SAP asks whether the user wishes to analyze a data base file. If the user responds with other than a carriage

ORIGINAL PAGE IS
OF POOR QUALITY

return, SAP proceeds with analysis; otherwise, SAP terminates. If the user specifies the /DB control switch during the source code file analysis stage, SAP automatically produces a project analysis for the current data base and project.

If no current data base file or project character exists, SAP prompts the user for this information. A complexity analysis report and a complexity correlation matrix are produced for each group of modules in the data base file with the project character specified by the user. The user terminates SAP execution by entering only a carriage return in response to the project character prompt.

2.3 SAP OUTPUT

SAP output consists of three listing files and two data files. The listing files and their contents are as follows:

FOR008.DAT - Module directory (/GB switch)
- Global summary (/GB switch)
- Project summary (/DB switch)

FOR007.DAT - Module statistics (/MO switch)
- Operator/operand summary (/HL switch)

FOR006.DAT - Source code listing (/LI switch)
- Symbol table dump (/DU switch)
- Error and warning messages

These files are described in Sections 2.3.1 through 2.3.3, respectively.

The SAP output data files are the data base and sequential output files discussed in Section 2.2.1 under the /DB and /SL control switches, respectively. The contents of these files are described in Sections 2.3.4 and 2.3.5, respectively.

For detailed information on all the statistics that appear in the listing files and the output data files, see Section 3.2. That section shows or references the method of calculation and lists the location of each statistic within the reports and files.

ORIGINAL PAGE IS
OF POOR QUALITY

2.3.1 GLOBAL STATISTICS FILE (FOR008.DAT)

The global statistics file contains the listings defined by the /GB and /DB switches. A new version of this file is created for each session with SAP in which output is directed to this file. The module directory, the global summary, and the project summary are described below.

2.3.1.1 Module Directory

Figure 2-1 is an example of the module directory. The output contains information on a module-by-module basis, as follows:

- Module counter
- Module name
- Two-letter descriptor of module type (BL (BLOCK DATA), MA (Main Program), FU (FUNCTION) and SU (SUBROUTINE))
- Number of source lines of code
- Number of comment lines
- Number of executable statements
- Number of nonexecutable statements
- Number of assignment statements
- Number of input/output statements
- Number of control statements
- Number of structure statements
- Number of INCLUDE statements
- Number of undecoded statements
- Total usage of all Halstead operators appearing in the module (Appendix B)

[illegible] 19-AUG-82 SOURCE ANALYZER PROGRAM V2 GLOBAL STATISTICS SUMMARY FILE LUN 8 ADDPOT PAGE 1

SAMPLE.FOR;20[illegible]

MODULE DIRECTORY

	MODULE NAME		SOURCE LINES	CNTS	EXEC STMT	N-EXC STMTS	STATEMENT INFORMATION COUNT ASGN	I/O	CNTL	STRU	INCL	UNDE	HALSTEAD OPTR	OPND	****** CYCLO	COMPLEXITIES SEL	USER1	USER2	****** PAGE NO.	NUMBER OF ERRS	WARN
1	ALPHCT	SM	87	65	9	9	5	1	7	3	0	0	26	23	2	92.0	0.0	0.0	1	0	0
2	ASCORD	SM	150	79	35	24	10	0	21	17	0	0	106	79	12	283.0	0.0	0.0	3	0	0
3	CHRONT	SM	200	93	70	35	30	4	18	23	0	0	204	172	20	576.0	0.0	0.0	5	0	0
4	CNTAPH	SM	91	70	8	10	5	0	3	2	0	0	22	22	3	95.0	0.0	0.0	7	0	0
5	CUFF	SM	126	92	46	14	30	5	5	13	0	0	148	129	7	420.0	0.0	0.0	9	0	0
6	CURSOR	SM	318	190	111	25	77	0	35	0	0	0	469	442	34	1036.5	0.0	0.0	11	0	0
7	CURSAV	SM	67	49	9	8	5	0	2	5	0	0	24	20	5	74.0	0.0	0.0	13	0	0
8	CURSRT	SM	269	176	55	24	49	0	7	0	0	0	207	212	6	679.5	0.0	0.0	15	0	0
9	DELFWD	SM	121	57	44	14	12	15	17	4	0	0	82	61	7	216.0	0.0	0.0	17	0	0
10	DLFSPC	SM	103	54	19	26	1	12	5	8	0	0	16	9	4	66.0	0.0	0.0	19	0	0
11	DSCAN	SM	189	102	61	27	38	0	12	26	0	0	174	145	18	506.0	0.0	0.0	21	0	0
12	EHARDJ	SM	52	45	3	3	2	0	1	0	0	0	6	6	1	33.0	0.0	0.0	23	0	0
13	EHRING	SM	108	59	19	28	3	5	5	13	0	0	36	25	8	100.5	0.0	0.0	25	0	0
14	ESTIM	SM	116	81	25	6	23	0	1	3	0	0	81	69	3	275.0	0.0	0.0	27	0	0
15	FIRUTT	SM	69	43	13	7	10	0	1	5	0	0	36	33	4	120.0	0.0	0.0	29	0	0
16	FLYARM	SM	245	178	41	14	20	0	18	6	0	0	104	86	16	355.0	0.0	0.0	31	0	0
17	FRNAME	SM	66	43	14	9	10	0	1	7	0	0	40	36	5	123.0	0.0	0.0	33	0	0
18	GARCOL	SM	137	90	27	19	16	1	5	13	0	0	95	85	10	231.0	0.0	0.0	35	0	0
19	GLTHC	SM	191	105	50	29	23	6	15	16	0	0	115	93	9	393.0	0.0	0.0	37	0	0
20	TLIST	SM	51	13	23	14	15	1	10	2	0	0	59	50	3	198.0	0.0	0.0	39	0	0

Figure 2-1. Module Directory

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

- Total usage of all Halstead operands appearing in the module (Appendix B)
- Cyclomatic complexity (Appendix C)
- SEL complexity (Appendix D)
- User's complexity number 1 (Appendix D)
- User's complexity number 2 (Appendix D)
- Page number for module statistics report in file FOR007.DAT
- Number of SAP errors (internal processing errors, such as table overflow)
- Number of SAP warnings (syntax errors in the module)

2.3.1.2 Global Summary

Figure 2-2 is an example of the global summary. The statistics are averages, sums, and/or maxima of the majority of the statistics appearing on the module statistics page (Section 2.3.2). For ease of comparison, the global summary page format is approximately the same as the module statistics page, although some statistics gathered for individual modules are not summarized on the global summary page. The global summary presents statistics for one input file.

2.3.1.3 Project Summary

Figure 2-3 presents an example of the project summary output produced by SAP in the project analysis stage (Section 2.2.2). The content of the columns is described below. References to Halstead's software science metrics use the symbols defined in Section 3.5. BLOCKDATA modules are not analyzed and do not appear in this report.

- Module counter
- Module name
- Program length (N)

SAMPLE.FOR/NL/DB/SI/XR

***** GLOBAL SUMMARY *****

GLOBAL MODULE	TOTAL MODULES	NUMBER MAINS	NUMBER SUBROUTINES	NUMBER FUNCTIONS	NUMBER BLOCKDATAS	SOURCE LINES	CODE LINES	COMMENT LINES
	20	0	20	0	0	2750	1106	1644

MODULE COMPLEXITY STATISTICS	LINES PER MODULE: TOTAL	CODE	COMMENT	PROLOGUE LENGTH	COMMENT LINES: EMBEDDED	BLANK
	318 MAX.	128 MAX.	190 MAX.	162 MAX.	43 MAX.	57 MAX.
	137.5 AVG.	55.3 AVG.	82.2 AVG.	67.3 AVG.	16.9 AVG.	26.5 AVG.

MODULE COMMUNICATION STATISTICS	ENTRY POINTS: 0 TOTAL PER MODULE	SUBR. CALLS: 38 TOTAL PER MODULE	FUNCT. REF.: 4 TOTAL PER MODULE	EXTERNAL NAMES DEFINED	EXTERNALLY DEFINED REFERENCES	ASF DEFINITIONS PER MODULE	ASF REFERENCES PER MODULE	ARG. LIST LENGTH IN REFERENCES TO SUBR/FUNCT.
	1 MAX	7 MAX	1 MAX	8 MAX	1 MAX	0 MAX	0 MAX	7 MAX
	0.0 AVG	1.9 AVG	0.1 AVG	2.5 AVG	0.1 AVG	0.0 AVG	0.0 AVG	2.9 AVG

STATEMENT CLASS COUNTERS	EXECUTABLE: CNT.	PCT.	STATEMENT	NON-EXECUTABLE: CNT.	PCT.	STATEMENT	MISCELLANEOUS: CNT.	PCT.	STATEMENT
	684	65.3	EXECUTABLE	363	34.7	NON-EXECUTABLE	0	0.0	NAMELIST
	393	37.6	ASSIGNMENT	40	3.8	SUBPROGRAM	10	1.0	DATA
	194	17.6	CONTROL	80	7.7	SPECIFICATION	0	0.0	ASF DEFINED
	166	15.9	STRUCTURED	80	7.7	TYPE SPECIF.	42	4.0	FORMAT
	50	4.8	I/O						

0	ASF REF.	393	ASSIGNMENT	0	ACCEPT	0	ASSIGN	0	BACKSPACE	0	BLOCKDATA
0	BASE	38	CALL	0	CHARACTER	3	CLOSE	58	COMMON	0	COMPLEX
14	CONTINUE	10	DATA	0	DECODE	0	DEFINEFILE	0	DELETE	5	DIMENSION
0	DOUBLECOMP	0	DOUBLEPREC	21	DOWHILE	13	DO	0	ELSEIF	27	ELSE
0	ENCODE	21	FUNCD	0	ENDFILE	49	ENDIF	20	END	0	ENTRY
13	EQUIVALENCE	0	EXTERNAL	0	FIND	42	FORMAT	0	*FUNCTION*	22	*GOTO*
48	.IF	43	IF	2	IMPLICIT	0	INCLUDE	0	INQUIRE	28	INTEGER
0	INTRINSIC	44	LOGICAL	0	NAMELIST	5	OPEN	0	PARAMETER	0	PAUSE
0	PRINT	0	PROGRAM	9	READ	7	REAL	34	RETURN	0	REWIND
0	REWRITE	0	SAVE	0	STOP	20	SUBROUTINE	0	THEN	0	TYPE
33	WRITE	0	UNDECODED	0	UNLOCK	0	VIRTUAL				

ORIGINAL PAGE IS OF POOR QUALITY

Figure 2-2. Global Summary (1 of 2)

CONTROL STATEMENT BREAKDOWN	IF STMTS PER MODULE MAX.	IF STMTS PER MODULE AVG.	BLOCKIF NESTING MAX.	BLOCKIF NESTING AVG.	GOTO STMTS PER MODULE MAX.	GOTO STMTS PER MODULE AVG.	DO STMTS PER MODULE MAX.	DO STMTS PER MODULE AVG.	DO LOOP NESTING DEPTH MAX.	DO LOOP NESTING DEPTH AVG.	STMTS PER DO LOOP MAX.	STMTS PER DO LOOP AVE.
	30	3.2	6	1.7	5	1.1	5	0.6	3	1.4	68	11.8

ASSIGNMENT STATEMENT BREAKDOWN	VARIABLES PER ASSIGNMENT		OPERATORS PER STATEMENT		SUBSCRIPT COMPLEXITY	
	2.3 AVG.	7 MAX.	0.6 AVG.	9 MAX.	3 MAX.	1.1 AVG.

SPECIFICATION STATEMENT BREAKDOWN	VARIABLES NAMED PER MODULE MAX.	VARIABLES NAMED PER MODULE AVG.	VARIABLES REFERENCED PER MODULE EXEC. STMTS MAX.	VARIABLES REFERENCED PER MODULE EXEC. STMTS AVG.	VARIABLES REFERENCED PER MODULE COMMONS MAX.	VARIABLES REFERENCED PER MODULE COMMONS AVG.	EQUIVALENCED NAMES PER MODULE MAX.	EQUIVALENCED NAMES PER MODULE AVG.	DIMENSIONS PER ARRAY MAX.	DIMENSIONS PER ARRAY AVG.	CHARACTERS PER VARIABLE MAX.	CHARACTERS PER VARIABLE AVG.
	180	52.2	56	18.8	181	40.8	12	1.7	2	1.1	8	5.4

Figure 2-2. Global Summary (2 of 2)

ORIGINAL PAGE IS
OF POOR QUALITY

11:12:53 19-AUG-82 SOURCE ANALYZER PROGRAM V2 COMPLEXITIES SUMMARY FILE LUN 8 PAGE 3

DBSAMP.DAT/PROJ 0

	MODULE NAME	LENGTH PROG	LENGTH PRED	VOLUME PROG	VOLUME POTENT	LEVEL PROGRM	LEVEL LANGUAG	PREDICTED EFFORT	PREDICTED TIME	PREDICTED BUGS	EXEC STMTS	NON-C LINES	TOTAL LINES	CYCL CMPL	UNIQUE OPTR	UNIQUE OPND	TOTAL OPTR	TOTAL OPND	I/O VARS
1	ADDPOL	49	71	215	28	0.1326	3.782	1623	0.0250	0	9	22	87	2	9	12	26	23	7
2	ASSGTD	185	216	1027	69	0.0676	4.699	15196	0.2345	0	35	71	150	12	17	30	106	79	15
3	CENPOT	374	338	2772	53	0.0235	1.250	96901	1.4954	0	70	107	200	20	22	44	204	172	12
4	CNTAPH	44	52	179	15	0.0862	1.338	2085	0.0322	0	8	21	91	3	7	10	22	22	4
5	COFF	277	204	1521	19	0.0179	0.256	117761	1.8173	0	46	74	126	7	16	29	148	129	5
6	COLORS	911	640	6078	325	0.0535	17.399	113615	1.7533	7	111	128	318	34	8	94	469	442	54
7	COMPAR	44	71	193	19	0.1017	1.998	1900	0.0293	0	9	18	67	5	12	9	24	20	5
8	COMPHF	410	333	2484	104	0.0419	4.356	59352	0.9159	0	55	93	269	6	6	55	207	212	21
9	DEFINE	143	178	766	19	0.0257	0.504	29868	0.4609	0	44	64	121	7	19	22	82	61	8
10	DESCPL	25	39	95	19	0.2065	1.057	461	0.0071	0	19	49	103	4	7	7	16	9	8
11	DSCAN	319	258	1815	84	0.0349	2.231	52659	0.8126	0	61	87	189	18	22	32	174	145	14
12	FRAPOT	12	14	38	11	0.3052	3.543	124	0.0019	0	3	7	52	1	4	5	6	6	3
13	FRPASG	61	102	290	64	0.2207	14.122	1314	0.0203	0	19	49	108	8	16	11	36	25	14
14	FSTIM	150	222	831	80	0.0969	7.818	8601	0.1327	0	25	35	116	3	12	35	81	69	17
15	FUNJIT	69	76	307	24	0.0780	1.872	3944	0.0609	0	13	20	63	4	10	12	36	33	6
16	FUNARI	190	204	1049	69	0.0662	4.601	15850	0.2446	0	41	67	245	16	20	26	104	86	15
17	FUNAME	76	80	348	24	0.0689	1.653	5059	0.0781	0	14	23	66	5	11	13	40	36	6
18	GARCOL	180	150	930	24	0.0258	0.619	36082	0.5568	0	27	47	137	10	17	19	95	85	6
19	GLEAF	289	197	1128	75	0.0665	4.992	16971	0.2619	0	50	86	191	9	16	27	115	93	16
20	TOTST	109	156	567	15	0.0271	0.424	20788	0.3208	0	23	38	81	3	16	21	59	50	4

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 2-3. Project Summary

ORIGINAL PAGE IS
OF POOR QUALITY

- Predicted program length ($\hat{N}$)
- Program volume (V)
- Potential program volume (V*)
- Program level (L)
- Language level (λ)
- Predicted effort (E)
- Predicted time ($\hat{T}$)
- Predicted bugs ($\hat{B}$)
- Number of executable statements
- Number of noncomment lines
- Total lines
- Cyclomatic complexity
- Unique operators (n_1)
- Unique operands (n_2)
- Total operators (N_1)
- Total operands (N_2)
- Number of input/output parameters (n_2^*)

Figure 2-4 shows an example of the correlation coefficients calculated from data in the project summary. The output is the matrix of correlation coefficients between the following seven variables, calculated by using the values of these variables for all modules presented within the preceding project summary:

- Actual program length (N)
- Predicted program length ($\hat{N}$)
- Executable statement count
- Noncomment source line count
- Total source line count
- Cyclomatic complexity
- Predicted effort (E)

A correlation coefficient report is produced for each project summary report.

1111/153 19-AUG-82 SOURCE ANALYZER PROGRAM V2 COMPLEXITIES SUMMARY FILE LUN 8

DBSAMP.DAT/PROJ 0

CORRELATION COEFFICIENTS FOR 20 MODULES

(ALEN) ACTUAL PROGRAM LENGTH
(PLEN) PREDICTED PROGRAM LENGTH
(EXEC) EXECUTABLE STATEMENT COUNT
(NCON) NON COMMENT SOURCE LINE COUNT
(LINE) TOTAL SOURCE LINE COUNT
(CYCL) CYCLOMATIC COMPLEXITY
(PEFF) PREDICTED EFFORT

MEASURE VS	ALEN	PLEN	EXEC	NCON	LINE	CYCL	PEFF
1 ALEN	1.00	0.98	0.95	0.86	0.85	0.87	0.82
2 PLEN	0.98	1.00	0.95	0.88	0.86	0.86	0.79
3 EXEC	0.95	0.95	1.00	0.96	0.88	0.89	0.83
4 NCON	0.86	0.88	0.96	1.00	0.89	0.82	0.79
5 LINE	0.85	0.86	0.88	0.89	1.00	0.80	0.63
6 CYCL	0.87	0.86	0.89	0.82	0.80	1.00	0.68
7 PEFF	0.82	0.79	0.83	0.79	0.63	0.68	1.00

Figure 2-4. Correlation Coefficients Matrix

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

2.3.2 MODULE STATISTICS FILE (FOR007.DAT)

Module output consists of the individual module statistics pages and the operator/operand summary pages. Figure 2-5 is an example of a module statistics page. The module statistics page is divided into eight "paragraphs". Each paragraph is labeled with a name appearing to the left (except for the fourth paragraph, Statement Type Counters, which is not labeled). The output of this report (and the paragraphs that appear in it) is controlled by the /MO switch and the auxiliary paragraph switches (Section 2.2.1, Table 2-1).

Figure 2-6 is an example of the operator/operand summary controlled by the /HL switch. The counts of the four types of operators (Appendix B) are presented in individual paragraphs. The summaries of delimiter and keyword operators list the counts of every possible operator. A zero count indicates the operator was not detected in the module. The procedure and transfer operators are listed only for those detected.

2.3.3 LISTING FILE (FOR006.DAT)

The listing file contains an echo of the input FORTRAN source code when the /LI switch is set. This file will also contain the symbol table dump produced when the /DU switch is set. The format and content of the symbol table dump are given in the SAP system description document (Reference 12).

SAP error and warning messages are directed to this file. Most SAP messages have the following format:

```
***** routine type ***** - msg
```

where routine is the name of the SAP module producing the message

type is the type of message (ERROR or WARNING)

msg is the narrative description of the error or warning

```
11:12:45 19-AUG-82 SOURCE ANALYZER PROGRAM V2   MODULE STATISTICS SUMMARY FILE          LUN  7    TDIST

SAMPLE.FOR/HI./DB/SI./XP
                                   MODULE TYPE = SUBROUTINE      MODULE NAME = TDIST

                        COMMUNICATIONS TO EXTERNAL
                    NAMES REFERENCED          TOTAL REFERENCES        ARGUMENT LIST LENGTH      COMMUNICATIONS
                                                                      TO SUBROUTINES AND        FROM EXTERNAL
EXTERNAL            3 SUBPROGRAM              3 SUBPROGRAMS             FUNCTIONS               0 ENTRY POINTS
COMMUNICATIONS      1 FUNCTIONS               1 FUNCTIONS                 2 MAX.                4 INPUT ARGUMENTS
                    .0 ASF DEF.               0 ASF DEF.                1.3 AVG.              0 ENTRY ARGUMENTS
                    0 EXT. NAMES              0 EXT. NAMES                                    2 RETURNS
                    0 NAMED COMMONS

MODULE                    COMMENT LINES          COMMENT PACKETS          CODE PACKETS          NUMBER OF LINES
COMMENTING          1 BLANK        4 PROLOGUE       8 COUNT                 13 COUNT              51 TOTAL
                    0 IN-LINE      9 EMBEDDED          SIZE(LINES)             SIZE(LINES)        38 CODE
                                                    1.1 AVG.   2 MAX.       2.9 AVG.   9 MAX.     13 COMMENT

                        EXECUTABLE                          NON-EXECUTABLE                      MISCELLANEOUS
                    CNT.  PCT. STATEMENT          CNT.  PCT. STATEMENT      CNT.  PCT. STATEMENT      CNT.  PCT. STATEMENT
STATEMENT           23  62.2 EXECUTABLE           14  37.8 NON-EXECUTABLE    0   0.0  NAMELIST
CLASS               15  40.5  ASSIGNMENT           2   5.4  SUBPROGRAM        0   0.0  DATA
COUNTERS            10  27.0  CONTROL              0   0.0  SPECIFICATION     0   0.0  ASF DEFINED      0   0.0  INCLUDE
                     2   5.4  STRUCTURED           6  16.2  TYPE SPECIF.      1   2.7  FORMAT           0   0.0  UNDECODED
                     1   2.7  I/O

                    0  ASF DEF.          15  ASSIGNMENT        0  ACCEPT          0  ASSIGN          0  BACKSPACE       0  BLOCKDATA
                    0  BYTE               3  CALL              0  CHARACTER       0  CLOSE           0  COMMON          0  COMPLEX
                    3  CONTINUE           0  DATA              0  DECODE          0  DEFINEFILE      0  DELETE          0  DIMENSION
                    0  DOUBLECOMP         0  DOUBLEPREC        0  DOWHILE         1  DO              0  ELSEIF          1  ELSE
                    0  ENCODE             0  ENDDO             0  ENDFILE         1  ENDIF           1  END             0  ENTRY
                    0  EQUIVALENCE        0  EXTERNAL          0  FIND            1  FORMAT          0  FUNCTION        0  GOTO
                    0  .IF                1  IF                0  IMPLICIT        0  INCLUDE         0  INQUIRE         2  INTEGER
                    0  INTRINSIC          1  LOGICAL           0  NAMELIST        0  OPEN            0  PARAMETER       0  PAUSE
                    0  PRINT              0  PROGRAM           0  READ            3  REAL            2  RETURN          0  REWIND
                    0  REWRITE            0  SAVE              0  STOP            1  SUBROUTINE      0  THEN            0  TYPE
                    1  WRITE              0  UNDECODED         0  UNLOCK          0  VIRTUAL
```

Figure 2-5. Module Statistics Page (1 of 2)

ORIGINAL PAGE IS
OF POOR QUALITY

	IF		GOTO	UNCOND. GOTO	MODULE CONTROL			I/O STMT	
	1 TOTAL	NESTING	0 TOTAL	0 OBJECT OF IF	0 PAUSE	1 DO LOOPS		BRANCH	TOTAL
CONTROL	0 LOGICAL	DEPTH	0 ASSIGNED	0 UPWARD	2 RETURN	NESTING	STATEMENTS	1 ERR=	2 BRANCHES
STATEMENTS	0 ARITH.	1.0 AVE	0 COMPUTED	0 DOWNWARD	0 RETURN I	DEPTH	PER LOOP	0 END=	2 TARGET
	1 BLOCK	1 MAX	0 UNCOND.		0 STOP	1.0 AVG.	19.0 AVG.		LABELS
	0 ELSE IF					1 MAX.	19 MAX.		

ASSIGNMENT STATEMENT	VARIABLES PER ASSIGNMENT 2.3 AVG. 4 MAX.	OPERATORS PER STATEMENT 0.5 AVG. 1 MAX.	SUBSCRIPT COMPLEXITY 1.0 AVG. 1 MAX.

	VARIABLE NAMES	REFERENCED VARIABLES	ARRAY DIMENSIONS	NUMBER OF CHARACTERS IN A VARIABLE NAME
SPECIFICATION	16 IN MODULE	15 IN CODE	1.0 AVG. 1 MAX.	2.1 AVG. 6 MAX.
STATEMENT	0 IN COMMON	0 IN COMMONS		
BREAKDOWN	0 EQUIVALENCED			

HALSTEAD ANALYSIS

COMPLEXITY ANALYSIS	TOTAL	UNIQUE	LEVEL	PREDICTED	SEL COMPLEXITY	CYCLOMATIC COMPLEXITY
	59 OPERATORS	16	0.03 PROGRAM	156 PROGRAM LENGTH	198.00	3
	50 OPERANDS	21	0.42 LANGUAGE	20788 EFFORT REQUIRED		

Figure 2-5. Module Statistics Page (2 of 2)

ORIGINAL PAGE IS OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

```
11:12:46 19-AUG-82 SOURCE ANALYZER PROGRAM V2   MODULE STATISTICS SUMMARY FILE          LUN  7    TDIST

SAMPLE.FOR/HI/DM/SI/KP

                                        HALSTEAD OPERATORS

DELIMETERS
           0  //        0  **        3  *        0  /          2  +        3  -
          15  =         5  (         2  ,        0  &          0  .NE.     0  .LT.
           0  .LE.      0  .EQ.      0  .GE.     0  .GT.       0  .AND.    0  .OR.
           0  .XOR.     0  .EQV.     0  .NOT.    0  .NEQV.
KEYWORDS
           0  IF()          0  IF(),,       1  .IF()          0  ELSE IF      1  ELSE        1  DO=,,
           0  DOWHILE       0  ASSIGNTO    20  EOS
PROCEDURES
           1  VERIFY        1  SORT         1  FKRMSG         1  ABORT
TRANSFERS
           1  FKR=       300
           1  ALT.RET.   FKRMSG 400

                                        HALSTEAD OPERANDS

           2 1          6 T          1 K          1 N          3 R          2 X          2 Y          1 27
           1 H2         2 X2         2 Y2         4 0.0        1 300        1 400        3 DX         3 DY
           3 XI         3 YL         2 EPR        3 DIST       2 MSGNUM
```

Figure 2-6. Operand/Operator Summary

ORIGINAL PAGE IS
OF POOR QUALITY

Appendix E lists all SAP error messages and discusses the most probable cause for each error.

2.3.4 DATA BASE FILE

The data base file contains a header record and record pairs for each module processed while the /DB switch is on. All records in the data base are formatted and are 80 bytes in length. This file is the source of data for the SAP project analysis stage (Section 2.2.2).

The header record contains an integer specifying the maximum number of records allowed in the file.

Two records are used to describe each module. Figure 2-7 is a listing of a sample data base file. The first record of each pair contains the following:

- Project code
- Module name

The second record of each pair contains the following:

- Number of arguments passed to the module
- Total number of COMMON block variables
- Count of blank comment lines
- Total count of comment lines
- Count of executable statements
- Count of external references (CALLs, function references, assignment statement function references)
- Count of input/output (ACCEPT, PRINT, READ, TYPE, and WRITE) statements
- Total number of source lines
- Count of unique Halstead operators
- Count of unique Halstead operands
- Total usage of Halstead operators

ORIGINAL PAGE IS
OF POOR QUALITY

```
 100
DADDPOT
  2 24 65 2    9 1 1  67  9 12  26  23  1  1  7
DASGNID
  3112 79 5  35 6 0 150 17 30 106  79 11 11 15
DCINPUT
  2 34 93 3  70 7 2 200 22 44 204 172 12 19 12
DCNTXFR
  0 32 70 2   8 0 0  91  7 10  22  22  0  2  4
DCOEF
  4 12 52 1  46 6 5 126 16 29 148 129  1  6  5
DCOLGLR
  0181190 8 111 0 0 318  8 94 469 442 30 33 54
DCOMPAR
  5 12 49 1   9 0 0  67 12  9  24  20  2  4  5
DCOMPWT
  0113176 7  55 0 0 269  6 55 207 212  0  5 21
DDEFINE
  2 12 57 1  44 510 121 19 22  82  61  5  6  5
DDFFSET
  0 14 54 2  19 011 103  7  7  16   9  3  3  5
DDSCAN
  1 30102 5  61 4 0 169 22 32 174 145 11 17 14
DERAPOT
  0 12 45 1   3 0 0  52  4  5   6   6  0  0  3
DERRMSG
  3 35 59 6  19 1 5 108 16 11  36  25  6  7 14
DESTIM
 17  0 81 0  25 5 0 116 12 35  81  69  1  2 17
DFINDIT
  6  0 43 0  13 0 0  63 10 12  36  33  1  3  6
DFLVART
  4178178 6  41 2 0 245 20 26 104  86 11 15 15
DFMNAME
  6  0 43 0  14 0 0  66 11 13  40  36  2  4  6
DGARCOL
  1 24 90 4  27 1 1 137 17 19  95  85  5  9  6
DGLINE
  8 42105 4  50 8 6 191 16 27 115  93  4  8 14
DIDIST
  4  0 13 0  23 4 1  51 16 21  59  50  1  2  4
```

Figure 2-7. Sample SAP Data Base File

ORIGINAL PAGE IS
OF POOR QUALITY

- Total usage of Halstead operands
- Count of IF (IF and .IF) statements
- Number of decisions
- Number of input/output parameters (module arguments, entry point arguments, and referenced COMMON block variables)

2.3.5 SEQUENTIAL OUTPUT FILE

The sequential file, ALL.SAP, contains a single formatted record for each module processed while the /SL switch is set. The current content and format of this file are based on the requirements specified for transferring data from SAP to the SEL software development data base component information files (Reference 13). Figure 2-8 is a listing of a sample sequential file. The following information is contained on each record:

- Project name
- Subsystem prefix
- Module name
- Number of arguments passed to the module
- Number of blank comment lines
- Number of executable statements
- Number of input/output (ACCEPT, PRINT, READ, TYPE, and WRITE) statements
- Total number of source lines
- Count of unique Halstead operators
- Count of unique Halstead operands
- Total usage of Halstead operators
- Total usage of Halstead operands

ORIGINAL PAGE IS
OF POOR QUALITY

RSAMP	SAADOPOT	2	65	9	1	87	9	12	26	23	1	1	7	5	0	0	2	4	5	1	1
RSAMP	SAASGVID	3	79	35	0	150	17	30	106	79	11	11	15	12	0	0	11	28	10	6	0
RSAMP	SACINPUT	2	93	70	2	200	22	44	204	172	12	19	12	10	0	0	17	27	39	7	4
RSAMP	SACITYP	0	70	8	0	91	7	10	22	22	0	2	4	4	0	0	2	0	5	0	0
RSAMP	SACOEF	4	52	46	5	126	16	29	148	129	1	6	5	1	0	2	12	20	30	4	5
RSAMP	SACOLGLB	0190		111	0	318	8	94	469	442	30	33	54	54	0	0	0	0	77	0	0
RSAMP	SACONPAR	5	49	9	0	67	12	9	24	20	2	4	5	0	0	0	4	0	5	0	0
RSAMP	SACONPUT	0176		55	0	269	6	55	207	212	0		21	21	0	0	0	0	49	0	0
RSAMP	SADEFINE	2	57	4410		121	19	22	82	61	5	4	5	3	0	0	2	13	12	5	10
RSAMP	SADFFSET	0	54	1911		103	7	7	16	9	3	3	5	5	0	0	5	0	1	0	11
RSAMP	SADSCAN	1102		61	0	189	22	32	174	145	11	17	14	13	0	1	19	12	38	3	0
RSAMP	SAERAPOT	0	45	3	0	52	4	5	6	6	0	0	3	3	0	0	0	0	2	0	0
RSAMP	SAERMSG	3	59	19	5	108	16	11	36	25	6	7	14	11	0	0	8	2	3	1	5
RSAMP	SAESTIM	17	81	25	0	116	12	35	81	69	1	2	17	0	0	5	2	5	23	0	0
RSAMP	SAFINDIT	6	43	13	0	63	10	17	36	33	1	3	6	0	0	0	4	0	10	0	0
RSAMP	SAFLVARI	4178		41	0	245	20	26	104	86	11	15	15	11	0	0	4	4	20	2	0
RSAMP	SAFNNAME	6	43	14	0	66	11	13	40	36	2	4	6	0	0	0	5	0	10	0	0
RSAMP	SAGARCOL	1	90	27	1	137	17	19	95	85	5	9	6	5	0	1	9	2	16	0	1
RSAMP	SAGLINE	8105		50	6	191	16	27	115	93	8	8	16	8	0	2	10	24	23	6	4
RSAMP	SATDIST	4	13	23	1	51	16	21	59	50	1	2	4	0	0	1	1	5	15	3	1

Figure 2-8. Sample Sequential Output File

ORIGINAL PAGE IS
OF POOR QUALITY

- Count of IF (IF and .IF) statements
- Number of decisions
- Number of input/output parameters (module arguments, entry point arguments, and referenced COMMON block variables)
- Number of referenced COMMON block variables
- Count of DO statements
- Count of function references
- Count of structure statements
- Count of arguments to CALL statements
- Count of assignment statements
- Count of CALL statements
- Count of FORMAT statements

ORIGINAL PAGE IS
OF POOR QUALITY

SECTION 3 - OVERVIEW OF SAP

3.1 SAP PROCESSING OVERVIEW

SAP statistics are gathered for individual modules, and overall statistics for a complete input file are accumulated for the global summary. For a given module, SAP processes one complete statement at a time as follows:

1. Read (and print if requested) a line of code, remove Hollerith and literal fields, compress remaining blanks, and append continuation cards
2. Perform an initial lexical analysis, identifying delimiters and separating the input line into tokens
3. Recognize assignment statements
4. Identify keyword statements
5. Perform specific statement type analysis and gather statistics
6. Process labels and identify the end of DO loop and block IF structures
7. If the current statement is a logical IF, repeat steps 3, 4, and 5 for the statement that is the object of the logical IF

In SAP, a token is defined as a string of one or more characters bounded by a predefined delimiter string or an end of line. All tokens in a module are entered into a symbol table using a chained hash access algorithm. This symbol table is initialized to empty before processing each module.

ORIGINAL PAGE IS
OF POOR QUALITY

During statement processing, one of the following four events occurs:

1. Normal statistics on the statement are gathered.
2. The statement type is not identified and the statement is marked as UNDECODED.
3. The statement type is identified, but due to a syntax error, complete statistics on the statement are not gathered. This type of error may affect only the statistics for the single statement or may affect the validity of the overall statistics for the current module (that is, results are uncertain). If warnings occur, they are indicated by a count in the warning column (for that module) in the module directory (Section 2.3.1.1.).
4. An internal SAP problem, such as symbol table overflow, occurs. In this case, the module processing, and possibly the entire input file processing, is terminated. If an internal SAP error occurs, it is indicated by a count in the error column (for that module) in the module directory. A symbol table dump can be used to determine the cause of the problem (Reference 12).

3.2 STATISTICS GATHERED

As stated, SAP gathers statistics on individual modules and on a global basis for an entire input file. The individual statistics are listed in Tables 3-1 through 3-18. The tables are grouped in pairs that correspond to the "paragraphs" on the Module Statistic page (Figure 2-5). For example, Tables 3-1 and 3-2 both describe the statistics appearing in the external communications paragraph of the module summary.

Table 3-1. Module Summary External Communications Paragraph

	STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
NAMES REFERENCED	1	NUMBER OF SUBROUTINE NAMES DEFINED	NUMBER OF DIFFERENT SUBROUTINE NAMES APPEARING IN CALL STATEMENTS
	2	NUMBER OF FUNCTION NAMES REFERENCED	NUMBER OF DIFFERENT FUNCTION NAMES APPEARING IN MODULE
	3	NUMBER OF ARITHMETIC STATEMENT FUNCTIONS DEFINED	DIRECT COUNT
	4	NUMBER OF EXTERNAL NAMES DEFINED	DIRECT COUNT
	5	NUMBER OF COMMON BLOCKS REFERENCED	DIRECT COUNT
TOTAL REFERENCES	6	NUMBER OF SUBROUTINE CALLS	DIRECT COUNT
	7	NUMBER OF FUNCTION REFERENCES	DIRECT COUNT
	8	NUMBER OF REFERENCES TO ARITHMETIC STATEMENT FUNCTIONS	DIRECT COUNT
	9	NUMBER OF REFERENCES TO EXTERNALLY DEFINED NAMES	NUMBER OF REFERENCES TO SUBROUTINE AND FUNCTION NAMES PASSED TO MODULE THROUGH ARGUMENT LIST
ARGUMENT LIST LENGTH	10	LENGTH OF ARGUMENT LISTS IN REFERENCES TO SUBROUTINES AND FUNCTIONS: MAXIMUM	NUMBER OF ITEMS APPEARING IN ARGUMENT LISTS NOT COUNTING SUBSCRIPTS OF ARGUMENTS OR FUNCTIONS WITHIN AN ARGUMENT
	11	AVERAGE	
FROM EXTERNAL	12	NUMBER OF ENTRY POINTS	DIRECT COUNT
	13	NUMBER OF ARGUMENTS TO MODULE	DIRECT COUNT
	14	NUMBER OF ARGUMENTS IN ALL ENTRY POINTS	COUNT NUMBER OF VARIABLES IN ENTRY ARGUMENT LISTS
	15	NUMBER OF RETURN STATEMENTS	DIRECT COUNT

8721/82

ORIGINAL PAGE IS
OF POOR QUALITY

Table 4-2. External Communications Statistics Locator

STATISTIC NUMBER	APPEARS IN						STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[1]	PROJECT SUMMARY	MODULE STATISTICS	DATA BASE	SEQUENTIAL FILE	
1				•			47
2				•			44
3		mx A		•			56
4		mx A		•			53
5				•		•	49
6		T mx A		•	2	•	128
7		T mx A		•	2	•	45
8		mx A		•	2		55
9		mx A		•			56
10		mx		•			234
11		A		•			213
12		T mx A		•			48
13				•	•	•	244
14				•			236
16				•			173

NOTES:

1. THIS REPORT IS A SUMMARY FOR ALL MODULES IN A FILE. THE STATISTICS MAY BE REPORTED AS TOTALS (T), MAXIMA (mx), AND/OR AVERAGE (A)
2. THE TOTAL OF THESE THREE STATISTICS APPEARS AS A COUNT OF EXTERNAL REFERENCES.

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-3. Module Summary Commenting Paragraph

	STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
COMMENT LINES	16	NUMBER OF BLANK COMMENT LINES	DIRECT COUNT
	17	SUM OF UN LINE; COMMENTS FOLLOWING AN ! ON THE SAME LINE AS A FORTRAN STATEMENT	DIRECT COUNT
	18	NUMBER OF COMMENTS IN PROLOG (COMMENTS APPEARING BEFORE FIRST EXECUTABLE STATEMENT IN MODULE)	DIRECT COUNT
	19	SUM OF ALL EMBEDDED (NONPROLOG) COMMENTS	DIRECT COUNT
COMMENT PACKETS	20	NUMBER OF COMMENT PACKETS	DIRECT COUNT
	21	LENGTH OF NONPROLOG COMMENT PACKETS AVERAGE	NUMBER OF COMMENT LINES IN COMMENT PACKETS APPEARING AFTER FIRST EXECUTABLE STATEMENT
	22	MAXIMUM	
CODE PACKETS	23	NUMBER OF CODE PACKETS	DIRECT COUNT
	24	NUMBER OF LINES BETWEEN COMMENT PACKETS. AVERAGE	NUMBER OF CODE LINES BETWEEN COMMENT PACKETS
	25	MAXIMUM	
NUMBER OF LINES	26	TOTAL NUMBER OF SOURCE LINES	DIRECT COUNT
	27	NUMBER OF CODE LINES	DIRECT COUNT
	28	NUMBER OF COMMENT LINES	DIRECT COUNT

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-4. Comment Statistics Locator

STATISTIC NUMBER	APPEARS IN						STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[1]	PROJECT SUMMARY	MODULE STATISTICS	DATA BASE	SEQUENTIAL FILE	
16		mx A		•	•	•	21
17				•			18
18		mx A		•			16
19		mx A		•			17
20				•			20
21				•			2
22				•			19
23				•			15
24				•			1
25				•			14
26	•	T mx A	•[2]	•	•	•	11
27		T mx A	•[2]	•			12
28	•	T mx A		•	•		13

NOTES.

1 THIS REPORT IS A SUMMARY FOR ALL MODULES IN A FILE. THE STATISTICS MAY BE REPORTED AS A TOTAL (T), MAXIMA (mx), AND/OR AVERAGE (A)

2 THESE STATISTICS ARE INCLUDED IN THE CORRELATION COEFFICIENT MATRIX.

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-5. Module Summary Statement Class Counter Paragraph

STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
	EXECUTABLE STATEMENTS.	THE METHOD OF CALCULATION FOR THE ENTRIES LISTED IN THIS TABLE IS BY DIRECT COUNT
29	NUMBER	
30	PERCENTAGE	
	ASSIGNMENT STATEMENTS (CLASS 1)	
31	NUMBER	
32	PERCENTAGE	
33	NUMBER AND PERCENTAGE OF CONTROL	
34	STATEMENTS (CLASS 2)	
35	NUMBER AND PERCENTAGE OF STRUCTURED	
36	STATEMENTS (CLASS 10)	
37	NUMBER AND PERCENTAGE OF INPUT/OUTPUT	
38	STATEMENTS (CLASS 7)	
39	NUMBER AND PERCENTAGE OF NONEXECUTABLE	
40	STATEMENTS	
41	NUMBER AND PERCENTAGE OF SUBPROGRAM	
42	STATEMENTS (CLASS 3)	
43	NUMBER AND PERCENTAGE OF SPECIFICATION	
44	STATEMENTS (CLASS 4)	
45	NUMBER AND PERCENTAGE OF TYPE SPECIFICATION	
46	STATEMENTS (CLASS 5)	
47	NUMBER AND PERCENTAGE OF NAMELIST	
48	STATEMENTS (CLASS 9)	
49	NUMBER AND PERCENTAGE OF DATA STATEMENTS	
50	(CLASS 6)	
51	NUMBER AND PERCENTAGE OF DEFINED ARITHMETIC	
52	STATEMENT FUNCTIONS (CLASS 13)	
53	NUMBER AND PERCENTAGE OF FORMAT	
54	STATEMENTS (CLASS 8)	
55	NUMBER AND PERCENTAGE OF INCLUDE	
56	STATEMENTS (CLASS 11)	
57	NUMBER AND PERCENTAGE OF UNDECODED	
58	STATEMENTS (CLASS 12)	

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-6. Statement Class Counter Locator

STATISTIC NUMBER	APPEARS IN						STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[1]	PROJECT SUMMARY	MODULE STATISTICS	DATA BASE	SEQUENTIAL FILE	
29	•	T	•[2]	•	•	•	73
30		T		•			71
31	•	T		•		•	101
32		T		•			81
33	•	T		•			102
34		T		•			82
35	•	T		•		•	110
36		T		•			90
37	•	T		•			107
38		T		•			87
39	•	T		•			74
40		T		•			72
41		T		•			103
42		T		•			83
43		T		•			104
44		T		•			84
45		T		•			106
46		T		•			85
47		T		•			109
48		T		•			89
49		T		•			108
50		T		•			86
51		T		•			99
52		T		•			85
53		T		•		•	108
54		T		•			88
55	•	T		•			111
56		T		•			91
57	•	T		•			112
58		T		•			92

NOTES:
1. THE COUNTS AND PERCENTAGES ARE FOR THE TOTAL (T) FILE
2. THE NUMBER OF EXECUTABLE STATEMENTS IS INCLUDED IN THE CORRELATION COEFFICIENT MATRIX.

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-7. Module Summary Statement Type Paragraph (1 of 2)

STATISTIC NUMBER	ENTRY (NUMBER OF STATEMENTS)	METHOD OF CALCULATION
59	ARITHMETIC STATEMENT FUNCTION DEFINITION	THE METHOD OF CALCULATION FOR THE ENTRIES LISTED IN THIS TABLE IS BY DIRECT COUNT.
60	ASSIGNMENT	
61	ACCEPT	
62	ASSIGN	
63	BACKSPACE	
64	BLOCKDATA	
65	BYTE	
66	CALL	
67	CHARACTER	
68	CLOSE	
69	COMMON	
70	COMPLEX	
71	CONTINUE	
72	DATA	
73	DECODE	
74	DEFINE FILE	
75	DELETE	
76	DIMENSION	
77	DOUBLE COMPLEX	
78	DOUBLE PRECISION	
79	DO WHILE	
80	DO	
81	ELSE IF	
82	ELSE	
83	ENCODE	
84	ENDDO	
85	ENDFILE	
86	ENDIF	
87	END	
88	ENTRY	
89	EQUIVALENCE	
90	EXTERNAL	

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-7. Module Summary Statement Type Paragraph (2 of 2)

STATISTIC NUMBER	ENTRY (NUMBER OF STATEMENTS)	METHOD OF CALCULATION
91	FIND	THE METHOD OF CALCULATION FOR THE ENTRIES LISTED IN THIS TABLE IS BY DIRECT COUNT
92	FORMAT	
93	FUNCTION	
94	GO TO	(INCLUDES COMPUTED, ASSIGNED, AND UNCONDITIONAL)
95	IF	(INCLUDES ARITHMETIC, BLOCK, AND LOGICAL)
96	IF	
97	IMPLICIT	
98	INCLUDE	
99	INQUIRE	
100	INTEGER	
101	INTRINSIC	
102	LOGICAL	
103	NAMELIST	
104	OPEN	
105	PARAMETER	
106	PAUSE	
107	PRINT	
108	PROGRAM	
109	READ	
110	REAL	
111	RETURN	
112	REWIND	
113	REWRITE	
114	SAVE	
115	STOP	
116	SUBROUTINE	
117	THEN	
118	TYPE	
119	WRITE	
120	UNDECODED	
121	UNLOCK	
122	VIRTUAL	

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-8. Statement Type Counter Locator (1 of 3)

STATISTIC NUMBER	APPEARS IN						STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[1]	PROJECT SUMMARY	MODULE STATISTICS	DATA BASE	SEQUENTIAL FILE	
59		T		•			121
60	•	T		•		•	122
61		T		•	2	2	123
62		T		•			124
63		T		•			125
64		T		•			126
65		T		•			127
66		T		•		•	128
67		T		•			129
68		T		•			130
69		T		•			131
70		T		•			132
71		T		•			133
72		T		•			134
73		T		•			136
74		T		•			136
75		T		•			137
76		T		•			138
77		T		•			139
78		T		•			140
79		T		•			141
80		T		•		•	142
81		T		•			143
82		T		•			144

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-8. Statement Type Counter Locator (2 of 3)

STATISTIC NUMBER	APPEARS IN						STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[1]	PROJECT SUMMARY	MODULE STATISTICS	DATA BASE	SEQUENTIAL FILE	
83		T		•			145
84		T		•			146
85		T		•			147
86		T		•			148
87		T		•			149
88		T		•			150
89		T		•			151
90		T		•			152
91		T		•			153
92		T		•		•	154
93		T		•			155
94		T		•			156
95		T		•	3	3	157
96		T		•	3	3	158
97		T		•			159
98	•	T		•			160
99		T		•			161
100		T		•			162
101		T		•			163
102		T		•			164
103		T		•			165
104		T		•			166
105		T		•			167
106		T		•			168

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-8. Statement Type Counter Locator (3 of 3)

STATISTIC NUMBER	APPEARS IN						STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[1]	PROJECT SUMMARY	MODULE STATISTICS	DATA BASE	SEQUENTIAL FILE	
107		T		•	2	2	169
108		T		•			170
109		T		•	2	2	171
110		T		•			172
111		T		•			173
112		T		•			174
113		T		•			176
114		T		•			176
115		T		•			177
116		T		•			178
117		T		•			179
118		T		•	2	2	180
119		T		•	2	2	181
120	•	T		•			182
121		T		•			183
122		T		•			184

NOTES:

1. THE COUNTS ARE FOR THE TOTAL (T) FILE
2. THESE STATISTICS ARE SUMMED TO OBTAIN THE INPUT/OUTPUT STATEMENT COUNT.
3. THESE STATISTICS ARE SUMMED TO OBTAIN THE IF STATEMENT COUNT.

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-9. Module Summary Control Statement Paragraph

	STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
IF	123	TOTAL NUMBER OF IF STATEMENTS	SUM OF LOGICAL, ARITHMETIC, BLOCK IF, AND ELSE IF STATEMENTS
	124	NUMBER OF LOGICAL IF STATEMENTS	DIRECT COUNT
	125	NUMBER OF ARITHMETIC IF STATEMENTS	DIRECT COUNT
	126	NUMBER OF BLOCK IF STATEMENTS	DIRECT COUNT OF BLOCK AND STRUCTURED IF
	127	NUMBER OF ELSE IF STATEMENTS	DIRECT COUNT
	128	AVERAGE NESTING DEPTH OF BLOCK IF STRUCTURES	SUM OF NESTED DEPTHS DIVIDED BY NUMBER OF BLOCK IF STATEMENTS
	129	MAXIMUM NESTING DEPTH OF IF STATEMENTS	DIRECT COUNT
GO TO	130	TOTAL NUMBER OF GO TO STATEMENTS	SUM OF ASSIGNED, COMPUTED, AND UNCONDITIONAL GO TO STATEMENTS
	131	NUMBER OF ASSIGNED GO TO STATEMENTS	DIRECT COUNT
	132	NUMBER OF COMPUTED GO TO STATEMENTS	DIRECT COUNT
	133	NUMBER OF UNCONDITIONAL GO TO STATEMENTS	DIRECT COUNT
UNCONDITIONAL GO TO	134	NUMBER OF GO TO STATEMENTS THAT ARE THE OBJECT OF A LOGICAL IF STATEMENT	DIRECT COUNT
	135	NUMBER OF UPWARD POINTED UNCONDITIONAL GO TO STATEMENTS	COUNT OF UNCONDITIONAL GO TO STATEMENTS TO A LABEL OF A STATEMENT THAT HAS BEEN PREVIOUSLY SCANNED
	136	NUMBER OF DOWNWARD POINTED UNCONDITIONAL GO TO STATEMENTS	COUNT OF UNCONDITIONAL GO TO STATEMENTS USING A LABEL NOT YET ENCOUNTERED
MODULE CONTROL	137	NUMBER OF PAUSE STATEMENTS	DIRECT COUNT
	138	NUMBER OF RETURN STATEMENTS	DIRECT COUNT (INCLUDES RETURN I)
	139	NUMBER OF RETURN I STATEMENTS	DIRECT COUNT
	140	NUMBER OF STOP STATEMENTS	DIRECT COUNT
	141	NUMBER OF DO STATEMENTS	DIRECT COUNT
		NESTING DEPTH OF DO LOOPS:	SUM OF NESTED DEPTHS DIVIDED BY NUMBER OF DO STATEMENTS
	142 AVERAGE 143 MAXIMUM		DIRECT COUNT
		NUMBER OF STATEMENTS PER DO LOOP:	SUM OF NUMBER OF STATEMENTS PER DO LOOP DIVIDED BY NUMBER OF DO STATEMENTS
	144 AVERAGE 145 MAXIMUM		DIRECT COUNT
INPUT/OUTPUT STATEMENTS	146	NUMBER OF ERR = KEYWORDS IN INPUT/OUTPUT STATEMENTS	DIRECT COUNT
	147	NUMBER OF END = KEYWORDS IN INPUT/OUTPUT STATEMENTS	DIRECT COUNT
MISCELLANEOUS	148	TOTAL NUMBER OF BRANCH STATEMENTS	COUNT OF IF AND GO TO STATEMENTS AND END = AND ERR = KEYWORDS
	149	TOTAL NUMBER OF TARGET LABELS	COUNT OF LABELS REFERENCED BY UNCONDITIONAL GO TO STATEMENTS AND ERR = AND END = KEYWORDS

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-10. Control Statement Locator

STATISTIC NUMBER	APPEARS IN						STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[1]	PROJECT SUMMARY	MODULE STATISTICS	DATA BASE	SEQUENTIAL FILE	
123		mx A		•			
124				•			221
125				•			222
126				•			255
127				•			256
128		A		•			
129		mx		•			258
130		mx A		•			
131				•			225
132				•			226
133				•			224
134				•			223
135				•			246
136				•			245
137				•			174
138				•			231
139				•			232
140				•			177
141		mx A		•		•	142
142		A		•			208
143		mx		•			248
144		A		•			207
145		mx		•			250
146				•			229
147				•			230
148				•			254
149				•			228

NOTE:

1 THIS REPORT IS A SUMMARY FOR ALL MODULES IN A FILE THE STATISTICS MAY BE REPORTED AS TOTALS (T), MAXIMA (mx), AND/OR AVERAGES (A)

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-11. Module Summary Assignment Statement Paragraph

	STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
VARIABLES PER ASSIGNMENT	150 151	NUMBER OF VARIABLES PER ASSIGNMENT: AVERAGE MAXIMUM	NUMBER OF VARIABLES IN ASSIGNMENT STATEMENTS
OPERATORS PER STATEMENT	152 153	NUMBER OF OPERATORS PER STATEMENT: AVERAGE MAXIMUM	NUMBER OF OPERATORS IN ASSIGNMENT STATEMENTS
SUBSCRIPT COMPLEXITY	154 155	SUBSCRIPT COMPLEXITY: AVERAGE MAXIMUM	NUMBER OF SUBSCRIPT PARENTHESIS PAIRS PLUS OPERATORS FOR ALL ARRAY SUBSCRIPTS IN MODULE

ORIGINAL PAGE IS OF POOR QUALITY

Table 3-12. Assignment Statement Statistics Locator

STATISTIC NUMBER	APPEARS IN						STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[1]	PROJECT SUMMARY	MODULE STATISTICS	DATA BASE	SEQUENTIAL FILE	
150		A		•			204
151		mx		•			241
152		A		•			206
153		mx		•			243
154		A		•			208
155		mx		•			253

NOTE:

1. THIS REPORT IS A SUMMARY FOR ALL MODULES IN A FILE. THE STATISTICS MAY BE REPORTED AS TOTALS (T), MAXIMA (mx), AND/OR AVERAGES (A).

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-13. Module Summary Specification Statement Breakdown Paragraph

	STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
VARIABLE NAMES	156	TOTAL NUMBER OF VARIABLES NAMED IN MODULE	DIRECT COUNT
	157	NUMBER OF VARIABLES NAMED IN COMMON BLOCK	DIRECT COUNT
	158	NUMBER OF VARIABLES EQUIVALENCED	NUMBER OF DISTINCT VARIABLE AND ARRAY NAMES THAT APPEAR IN EQUIVALENCE STATEMENTS
REFERENCED VARIABLES	159	NUMBER OF VARIABLES REFERENCED IN CODE	COUNT OF ALL VARIABLE AND ARRAY REFERENCES
	160	NUMBER OF COMMON BLOCK VARIABLES REFERENCED	NUMBER OF DISTINCT VARIABLE AND ARRAY NAMES (DEFINED IN COMMON) THAT APPEAR IN EXECUTABLE STATEMENTS
ARRAY DIMENSIONS	161 162	NUMBER OF DIMENSIONS FOR ARRAYS: AVERAGE MAXIMUM	DIRECT COUNT
NUMBER OF CHARACTERS	163 164	NUMBER OF CHARACTERS PER VARIABLE NAME: AVERAGE MAXIMUM	DIRECT COUNT

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-14. Specification Statement Statistics Locator

STATISTIC NUMBER	APPEARS IN						STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[1]	PROJECT SUMMARY	MODULE STATISTICS	DATA BASE	SEQUENTIAL FILE	
156		mx A		•			43
157				•	•		50
158		mx A		•			58
159		mx A		•			57
160		mx A		•			51
161		A		•			32
162		mx		•			60
163		A		•			31
164		mx		•			41

NOTE

1 THIS REPORT IS A SUMMARY FOR ALL MODULES IN A FILE THE STATISTICS MAY BE REPORTED AS TOTALS (T), MAXIMA (mx), AND/OR AVERAGES (A)

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-15. Module Summary Complexity Analysis Paragraph

	STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
BASIC HALSTEAD MEASURES	165	TOTAL NUMBER OF OPERATORS	SEE APPENDIX B
	166	TOTAL NUMBER OF OPERANDS	SEE APPENDIX B
	167	NUMBER OF UNIQUE OPERATORS	SEE APPENDIX B
	168	NUMBER OF UNIQUE OPERANDS	SEE APPENDIX B
DERIVED HALSTEAD QUANTITIES	169	PROGRAM LEVEL	SEE SECTION 3.5
	170	LANGUAGE LEVEL	SEE SECTION 3.5
	171	PREDICTED PROGRAM LENGTH	SEE SECTION 3.5
	172	PREDICTED EFFORT REQUIRED	SEE SECTION 3.5
OTHER COMPLEXITIES	173	SEL WEIGHTED COMPLEXITY	SEE SECTION 3.4, APPENDIX D
	174	CYCLOMATIC COMPLEXITY	SEE SECTION 3.6, APPENDIX C

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-16. Complexity Analysis Statistic Locator

STATISTIC NUMBER	APPEARS IN						STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY	PROJECT SUMMARY	MODULE STATISTICS	DATA BASE	SEQUENTIAL FILE	
165	•		•	•	•	•	
166	•		•	•	•	•	
167			•	•	•	•	
168			•	•	•	•	
169			•	•			
170			•	•			
171			•[1]	•			
172			•[1]	•			
173	•			•			
174	•		•[1]	•			

NOTE

1 THESE STATISTICS ARE INCLUDED IN THE CORRELATION COEFFICIENT MATRIX.

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-17. Miscellaneous Statistics (Not Reported on the Module Statistics Page)

STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
175	NUMBER OF DECISIONS	SEE APPENDIX C
176	NUMBER OF INPUT/OUTPUT PARAMETERS TO THE MODULE	SEE APPENDIX B
177	USER COMPLEXITY NUMBER 1	SEE APPENDIX D
178	USER COMPLEXITY NUMBER 2	SEE APPENDIX D
179	ACTUAL PROGRAM LENGTH	SEE SECTION 3.5
180	PROGRAM VOLUME	SEE SECTION 3.5
181	POTENTIAL PROGRAM VOLUME	SEE SECTION 3.5
182	PREDICTED TIME	SEE SECTION 3.5
183	PREDICTED BUGS	SEE SECTION 3.5

Table 3-18. Miscellaneous Statistics Locator

STATISTIC NUMBER	APPEARS IN						STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY	PROJECT SUMMARY	MODULE STATISTICS	DATA BASE	SEQUENTIAL FILE	
175					•	•	
176			•		•	•	
177	•						
178	•						
179			•[1]				
180			•				
181			•				
182			•				
183			•				

NOTE.
1. THIS STATISTIC IS INCLUDED IN THE CORRELATION COEFFICIENT MATRIX.

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

The first table in each pair describes the statistic and shows or references the method of calculation used to obtain the statistic. The first column in this table (Statistic Number) represents an arbitrary sequential ordering of all the statistics and is used only to provide a reference to the second table in the pair.

The second table in the pair shows the reports and data files in which the statistic appears. The reports and files listed are the module directory (Figure 2-1), the global summary (Figure 2-2), the project summary (Figure 2-3), the module statistics (Figure 2-5), the data base file (Figure 2-7), and the sequential file (Figure 2-8). The last column shows the statistical weight index for the particular statistic (Section 3.4 and Appendix D).

As mentioned, the tables appear in pairs that correspond to the eight paragraphs on the Module Statistics page. The ninth pair of tables (Tables 3-17 and 3-18) lists statistics that do not appear on the Module Statistics page.

The tables refer to counts of lines and statements and to modules. The terms used in the tables are defined below.

- Line. A line is equivalent to a record in the input file. Comments are always counted in units of lines.

- Statement. A statement is composed of an initial line and any continuation lines that together contain a single FORTRAN statement. SAP makes one exception to this usual definition of a statement: a logical IF and the object statement of the logical IF are counted as separate statements (one each). Each statement type is classified as executable or nonexecutable. This classification is not the same as the classification used in the FORTRAN 77 standard and serves a different purpose than that classification. Tables A-1 through A-4 in Appendix A contain more information on statement classifications.

ORIGINAL PAGE IS
OF POOR QUALITY

- Module. A module is the collection of statements preceding and including an END statement. Modules are classified as BLOCKDATA, SUBROUTINE, FUNCTION, or main program.

3.3 KEYWORDS FILE

To allow flexibility in classifying statements and in marking statements executable or nonexecutable, an external keywords file is used. The keywords file location on the VAX-11/780 is DBB1:[TOOLS]KEYWORDS.SAP, and on the PDP-11/70, its location is DB1:[213,2]KEYWORDS.SAP.

Figure 3-1 is a listing of the keywords file as it is currently implemented. Column 1 is a logical constant, which, if true(T), indicates the statement is executable, and if false(F), nonexecutable. Columns 2 and 3 are obsolete and do not affect the execution of SAP. Column 4 contains the statement class. Column 5 contains the number of characters in the keyword. Column 6 is the keyword. The format for a keywords file record is (L3, 6X, 2I3, 1X, 16A1).

Table 3-19 shows the definition of each statement class.

3.4 STATISTICAL WEIGHTS FILE

The statistical weights file is used in determining the SEL figure of complexity for each module. The formula used to determine this complexity is

$$\text{SEL complexity} = \sum_{i=1}^{256} (\text{module statistic})_i \times (\text{statistic weight})_i$$

The SEL complexity is reported in the module directory (Section 2.3.1.1) and on the module summary page (Section 2.3.2).

Most statistics printed on the module summary page have been assigned a statistical weight index as shown in Tables 3-1

ORIGINAL PAGE IS
OF POOR QUALITY

```
F  0  0 13  9 ASF DEF.
T  0  0  1 10 ASSIGNMENT
T  0  0  7  6 ACCEPT
T  0  0  2  6 ASSIGN
T  0  0  7  9 BACKSPACE
F  0  0  3  9 BLOCKDATA
F  1     5  4 BYTE
T  0  0  2  4 CALL
F  0  0  5  9 CHARACTER
T  1  0  7  5 CLOSE
F  0  0  4  6 COMMON
F  0  0  5  7 COMPLEX
F  0  0  2  8 CONTINUE
F  0  0  6  4 DATA
T  1  0  7  6 DECODE
F  0  0  7 10 DEFINEFILE
T  1  0  7  6 DELETE
F  0  0  4  9 DIMENSION
F  1  0  5 13 DOUBLECOMPLEX
F  0  0  5 15 DOUBLEPRECISION
T  0  0 10  7 DOWHILE
T  0  0  2  2 DO
T  0  0 10  6 ELSEIF
F  0  0 10  4 ELSE
T  1  0  7  6 ENCODE
F  0  0 10  6 ENDDO
T  0  0  7  7 ENDFILE
F  0  0 10  5 ENDIF
F  0  0  3  3 END
F  0  0  3  5 ENTRY
F  0  0  4 11 EQUIVALENCE
F  0  0  4  8 EXTERNAL
T  0  0  7  4 FIND
```

Figure 3-1. Keywords File (1 of 2)

ORIGINAL PAGE IS
OF POOR QUALITY

```
F  0  0  8  6 FORMAT
F  0  0  3  8 FUNCTION
T  0  0  2  4 GOTO
T  0  0 10  3 .IF
T  0  0  2  2 IF
F  0  0  5  8 IMPLICIT
F  1  0 11  7 INCLUDE
T  0  0  7  7 INQUIRE
F  0  0  5  7 INTEGER
F  0  0  4  9 INTRINSIC
F  0  0  5  7 LOGICAL
F  0  1  9  8 NAMELIST
T  1  0  7  4 OPEN
F  1  0  4  9 PARAMETER
T  0  0  2  5 PAUSE
T  0  0  7  5 PRINT
F  1  0  3  7 PROGRAM
T  0  0  7  4 READ
F  0  0  5  4 REAL
T  0  0  2  6 RETURN
T  0  0  7  6 REWIND
T  1  0  7  7 REWRITE
F  0  0  4  4 SAVE
T  0  0  2  4 STOP
F  0  0  3 10 SUBROUTINE
F  0  0 10  4 THEN
T  0  0  7  4 TYPE
T  0  0  7  5 WRITE
F  0  0 12  9 UNDECODED
T  1  0  7  6 UNLOCK
F  1  0  4  7 VIRTUAL
```

Figure 3-1. Keywords File (2 of 2)

ORIGINAL PAGE IS
OF POOR QUALITY

Table 3-19. Statement Class Definitions

Statement Class	Statement Definition
1	Assignment
2	Control
3	Subprogram
4	Specification
5	Type Specification
6	DATA
7	Input/Output
8	FORMAT
9	NAMELIST
10	Structure
11	INCLUDE
12	Undecoded
13	Arithmetic Statement Function Definition

ORIGINAL PAGE IS
OF POOR QUALITY

through 3-18. The statistical weights file contains a weight to be associated with the statistics having the indicated range of indexes. An example of a statistical weights file is the default file shown in Figure 3-2. The default statistical weights file location on the VAX-11/780 is DBB1:[TOOLS]WEIGHTS.SAP and on the PDP-11/70, DB1:[213,2]WEIGHTS.SAP.

The user may specify another statistical weights file by using the /UW control switch (Section 2.2.1). If this switch is specified, SAP will prompt the user for the name of an alternative statistical weights file before processing the specified input file. The format for a statistical weights file is (2I5,F6.1). The remainder of each record may be used for comments. The first two columns specify the range of statistical weight indexes to be assigned the statistical weight (column 3). The weights are assigned to each specified range in the order in which the records appear in the file. For example, the first record in the default statistical weights file (Figure 3-2) sets all the weights (numbers 1 through 256) to an initial value of 0. The second record assigns a weight of 1.0 to the number of code lines in a module and so forth. The statistical weight indexes for particular statistics can be found in the locator table pairs (Tables 3-1 through 3-18). The statistical weights file is also discussed in Appendix D.

3.5 HALSTEAD'S SOFTWARE SCIENCE METRICS

SAP calculates several of the quantities defined by Halstead (Reference 9). The quantities are all based on the following five measures:

n_1 = number of unique or distinct operators appearing in the module

n_2 = number of unique or distinct operands appearing in the module

ORIGINAL PAGE IS
OF POOR QUALITY

```
  1  256     0.0  INITIALIZATION OF ALL WEIGHTS TO ZERO
 12   12     1.0  CODE LINES
 50   50     0.5  VARIABLES IN COMMON
 58   58     4.0  VARIABLES EQUIVALENCED
122  122    10.0  ASSIGNMENT STATEMENTS
167  167     2.0  PARAMETER STATEMENTS
233  233     2.0  TOTAL NUMBER OF ARGUMENTS IN CALL STATEMENTS
246  246     2.0  UPWARD UNCONDITIONAL GOTO STATEMENTS
```

Figure 3-2. Default Statistical Weights File

ORIGINAL PAGE IS
OF POOR QUALITY

N_1 = total usage of all operators appearing in the module

N_2 = total usage of all operands appearing in the module

n_2^* = number of unique input/output parameters to the module

Appendix B presents the details of the methods used by SAP to collect these five measures.

The quantities calculated by SAP and the formulas used are as follows:

Program length	$N = N_1 + N_2$
Predicted program length	$\hat{N} = n_1 \times \log_2 n_1 + n_2 \times \log_2 n_2$
Program volume	$V = N \times \log_2 (n_1 + n_2)$
Potential program volume	$V^* = (2 + n_2^*) \times \log_2 (2 + n_2^*)$
Program level	$L = V^*/V$
Language level	$\lambda = V^{*2}/V$
Predicted effort	$E = V/L$
Predicted time	$\hat{T} = E/S$
Predicted bugs	$\hat{B} = V/E_0$

where S is the Stroud number (Reference 14) (64,800 mental discriminations per hour) and E_0 is the mean error rate (3,000 discriminations between potential programming errors).

The module summary page (Section 2.3.2) presents the counts of unique operators and operands, the total counts of operators and operands, the program and language level, and the predicted program length and effort required. The project summary (Section 2.3.1.3) also presents the count of input/output parameters, the program volume and the potential volume, and the predicted time and bugs. The module directory (Section 2.3.1.1) presents the total usage of operators and operands.

ORIGINAL PAGE IS
OF POOR QUALITY

The /HL control switch (Section 2.2.1) can be used to present a detailed listing of the particular operators and operands detected by SAP.

3.6 McCABE'S COMPLEXITY MEASURE

SAP computes the cyclomatic complexity of a software module. According to McCabe (Reference 10), the cyclomatic complexity is calculated from the following formula:

Cyclomatic Complexity: $V = d + 1$

where d is the number of decisions in the module.

Appendix C presents the details of the methods used to count the number of decisions.

The cyclomatic complexity appears in the module directory (Section 2.3.1.1), the project summary (Section 2.3.1.3), and the module summary (Section 2.3.2).

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX A - FORTRAN STATEMENT ANALYSIS

A.1 DISCUSSION

This appendix presents information about how SAP processes each FORTRAN statement type. Tables A-1 through A-4 present an overview of this information.

Each table indicates for each statement type whether it is acceptable to standard FORTRAN (FORTRAN 77, Reference 1), DEC FORTRAN (References 2 through 5), or to IBM FORTRAN (References 6 and 7). Several Structured FORTRAN (SFORT) statements are also available on the DEC and IBM computers through the use of a preprocessor (Reference 8). The NAMELIST statement is also available on DEC computers through the use of the NAMELIST Preprocessor Program (NPP).

The tables also indicate how SAP and FORTRAN 77 classify each statement as executable or nonexecutable. This definition of "executable" for SAP is based on the contents of the keywords file as described in Section 3.3 and shown in Figure 3-1.

The last six columns of each table indicate how each statement is processed during the SAP analysis of Halstead operands and operators. An explanation of the terms used in the column headings in these tables is given in Appendix B.

The following section presents the individual statements recognized by SAP. The syntax for each statement is presented in the same format as it is presented in the FORTRAN standard (Reference 1), if the statement is acceptable to the standard. If the statement is not acceptable to all dialects, the dialect(s) to which it belongs is indicated (F77, DEC, IBM, SFORT, or NPP).

Unless specifically noted, SAP scans all delimiters and tokens in each statement.

ORIGINAL PAGE IS
OF POOR QUALITY

Table A-1. Assignment/Control Statement Type Summary

STATEMENT TYPE	FORTRAN 77	DEC	IBM	EXECUTABLE-SAP	EXECUTABLE-F77	OPERAND ANALYSIS	OPERATOR ANALYSIS				KEYWORD (EOS)
							DELIMITER	KEYWORD	PROCEDURE	TRANSFER	
ASSIGNMENT, ARITHMETIC	●	●	●	●	●	●	●		●		●
ASSIGNMENT, LOGICAL	●	●	●	●	●	●	●		●		●
ASSIGNMENT, CHARACTER	●	●		●	●	●	●		●		●
ASSIGN	●	●	●	●	●			●			●
CALL	●	●	●	●	●	●	●		●	●	●
DOWHILE		●	1	●		●	●	●	●		●
DO	●	●	●	●	●	●		●			●
ELSEIF	●	●	1	●	●	●	●	●	●		●
GOTO, UNCONDITIONAL	●	●	●	●	●	●				●	●
GOTO, COMPUTED	●	●	●	●	●	●	●			●	●
GOTO, ASSIGNED	●	●	●	●	●	●				●	●
IF, ARITHMETIC	●	●	●	●	●	●	●	●	●		●
IF, BLOCK	●	●		●	●	●	●	●	●		●
IF, LOGICAL	●	●	●	●	●	●	●	●	●		●
IF, STRUCTURED		2	1	●		●	●	●	●		●
PAUSE	●	●	●	●	●						
RETURN	●	●	●	●	●						
STOP	●	●	●	●	●						
CONTINUE	●	●	●		●						
ELSE	●	●	1		●						
ENDDO		●	1								
ENDIF	●	●	1		●						
THEN											

08/17/82

[1] WITH IBM VERSION OF SFORT.

[2] WITH DEC VERSION OF SFORT.

ORIGINAL PAGE IS
OF POOR QUALITY

Table A-2. Input/Output Statement Type Summary

STATEMENT TYPE	FORTRAN 77	DEC	IBM	EXECUTABLE-SAP	EXECUTABLE-F77	OPERAND ANALYSIS[1]	OPERATOR ANALYSIS				KEYWORD (EOS)[3]
							DELIMITER	KEYWORD	PROCEDURE	TRANSFER[2]	
ACCEPT		●		●		●				●	●
BACKSPACE	●	●	●	●	●	●				●	●
CLOSE	●	●		●	●	●				●	●
DECODE		●		●		●				●	●
DEFINEFILE		●	●			●				●	●
DELETE		●		●		●				●	●
ENCODE		●		●		●				●	●
ENDFILE	●	●	●	●	●	●				●	●
FIND		●	●	●		●				●	●
INQUIRE	●	●		●	●	●				●	●
OPEN	●	●		●	●	●				●	●
PRINT	●	●	●	●	●	●				●	●
READ	●	●	●	●	●	●				●	●
REWIND	●	●	●	●	●	●				●	●
REWRITE		●		●		●				●	●
TYPE		●		●		●				●	●
UNLOCK		●		●		●				●	●
WRITE	●	●	●	●	●	●				●	●
FORMAT	●	●	●								
NAMELIST		4	●								

[1]ERR = OR END = STATEMENT LABELS ONLY

[2]ERR = OR END = IF FOUND.

[3]COUNTED ONLY IF AN ERR = OR END = IS FOUND

[4]WITH NAMELIST PREPROCESSOR PROGRAM.

ORIGINAL PAGE IS
OF POOR QUALITY

Table A-3. Specification/Typing Statement Type Summary

STATEMENT TYPE	FORTRAN 77	DEC	IBM	EXECUTABLE SAP	EXECUTABLE F77	OPERAND ANALYSIS	OPERATOR ANALYSIS				KEYWORD (EQUS)
							DELIMITER	KEYWORD	PROCEDURE	TRANSFER	
COMMON	•	•	•								
DATA	•	•	•								
DIMENSION	•	•	•								
EQUIVALENCE	•	•	•								
EXTERNAL	•	•	•								
INTRINSIC	•	•									
PARAMETER	•	•									
SAVE	•	•									
VIRTUAL		•									
BYTE		•									
CHARACTER	•	•									
COMPLEX	•	•	•								
DOUBLECOMPLEX		•									
DOUBLEPRECISION	•	•	•								
IMPLICIT	•	•	•								
INTEGER	•	•	•								
LOGICAL	•	•	•								
REAL	•	•	•								

ORIGINAL PAGE IS
OF POOR QUALITY

Table A-4. Subprogram and Other Statement Type Summary

STATEMENT TYPE	FORTRAN 77	DEC	IBM	EXECUTABLE-SAP	EXECUTABLE-F77	OPERAND ANALYSIS	OPERATOR ANALYSIS				KEYWORD (EOS)
							DELIMITER	KEYWORD	PROCEDURE	TRANSFER	
ARITHMETIC STATEMENT FUNCTION DEFINITION	●	●	●								
BLOCKDATA	●	●	●								
END	●	●	●		●	●	●		●		●
ENTRY	●	●	●								
FUNCTION	●	●	●								
PROGRAM	●	●									
SUBROUTINE	●	●	●								
INCLUDE		●									
UNDECODED											

8/21/82

ORIGINAL PAGE IS
OF POOR QUALITY

A.2 FORTRAN STATEMENTS

A.2.1 ACCEPT STATEMENT

Syntax (DEC):

```
    ACCEPT   f[,list]
or  ACCEPT   *[,list]
```

A.2.2 ARITHMETIC STATEMENT FUNCTION DEFINITION

Syntax:

```
    fun ([d[,d]...]) = e
```

The defining reference to fun is counted as one reference to a Halstead procedure operator.

A.2.3 ASSIGN STATEMENT

Syntax:

```
    ASSIGN   s   TO   i
```

The statement is not parsed beyond statement label s.

A.2.4 ASSIGNMENT STATEMENT

Syntax:

```
    v = e
```

No distinction is made between arithmetic, logical, or character assignment statements.

A.2.5 BACKSPACE STATEMENT

Syntax:

```
    BACKSPACE   u
```

Syntax (also for DEC, F77):

```
    BACKSPACE   (alist)
```

ORIGINAL PAGE IS
OF POOR QUALITY

where alist consists of selections from the following

```
[UNIT=] u
IOSTAT = ios
ERR = s
```

The statement label, s, is the only item searched for in alist by SAP.

A.2.6 BLOCK DATA STATEMENT

Syntax:

```
BLOCKDATA   [sub]
```

A.2.7 BYTE STATEMENT

Syntax (DEC):

```
BYTE v[/clist/][,v[/clist/]...
```

Statement parsing is limited to marking variable names as array or nonarray. All names are marked as numeric.

A.2.8 CALL STATEMENT

Syntax:

```
CALL suo [([a[,a]...])]
```

A.2.9 CHARACTER STATEMENT

Syntax (DEC, F77):

```
CHARACTER[*len[,]] nam[,nam]...
```

where nam is one of the following

```
v[*len]
a[(d)][*len]
```

SAP marks variable names as array or nonarray.
All variable names are marked as type CHARACTER.

ORIGINAL PAGE IS
OF POOR QUALITY

A.2.10 CLOSE STATEMENT

Syntax (DEC, F77):

```
CLOSE (cllist)
```

where cllist consists of selections from the following

```
[UNIT=] u
IOSTAT = ios
ERR = s
STATUS = sta
```

The statement label, s, is the only item searched for in cllist by SAP.

A.2.11 COMMON STATEMENT

Syntax:

```
COMMON [/[cb]/] nlist [[,]/[cb]/nlist]...
```

SAP tags each cb as a COMMON name and tags each name in each nlist as a COMMON variable name. All names are marked as array or nonarray.

A.2.12 COMPLEX STATEMENT

Syntax (F77):

```
COMPLEX v[,v]...
```

Syntax (DEC, IBM)

```
COMPLEX[*n] v[/clist/][[,]v[/clist/]...
```

Statement parsing is limited to flagging variable names as array or nonarray. All names are marked as numeric.

A.2.13 CONTINUE STATEMENT

Syntax:

```
CONTINUE
```

No processing is done by SAP beyond the CONTINUE keyword.

ORIGINAL PAGE IS
OF POOR QUALITY

A.2.14 DATA STATEMENT

Syntax:

```
DATA nlist /clist/[[,]nlist/clist/]...
```

No processing is done by SAP beyond the DATA keyword.

A.2.15 DECODE STATEMENT

Syntax (DEC):

```
DECODE (c,f,b[,IOSTAT=ios][,ERR=s]) list
```

The statement label, s, is the only item searched for inside the parentheses.

A.2.16 DEFINEFILE STATEMENT

Syntax (DEC):

```
DEFINEFILE u(m,n,U,v)[,u(m,n,U,v)]...
```

Syntax (IBM)

```
DEFINEFILE u(m,n,f,v)[,u(m,n,f,v)]...
```

A.2.17 DELETE STATEMENT

Syntax (DEC):

```
DELETE([UNIT=]u[,REC=r][,IOSTAT=ios][,ERR=s])
```

or `DELETE(u'r[,IOSTAT=ios][,ERR=s])`

The statement label, s, is the only item searched for inside the parentheses.

A.2.18 DIMENSION STATEMENT

Syntax:

```
DIMENSION a(d)[,a(d)]...
```

All names are flagged as arrays.

ORIGINAL PAGE IS
OF POOR QUALITY

A.2.19 DO STATEMENT

Syntax (F77):

DO s[,]i = e_1, e_2[,e_3]

Syntax (IBM):

DO s i = e_1, e_2[,e_3]

Syntax (DEC):

DO [s[,]]i = e_1, e_2[,e_3]

Processing of this statement includes loop nesting calculations.

A.2.20 DOUBLECOMPLEX STATEMENT

Syntax (DEC):

DOUBLECOMPLEX v[/clist/][,v[/clist/]...

Statement parsing is limited to marking variable names as array or nonarray. All names are marked as numeric.

A.2.21 DOUBLEPRECISION STATEMENT

Syntax (F77):

DOUBLEPRECISION v[,v]...

Syntax (DEC, IBM):

DOUBLEPRECISION v[/clist/][,v[/clist]...

Statement parsing is limited to marking variable names as array or nonarray. All names are marked as numeric.

A.2.22 DOWHILE STATEMENT

Syntax (DEC):

DO [s[,]] WHILE (e)

Syntax (SFORT):

DOWHILE(e)

ORIGINAL PAGE IS
OF POOR QUALITY

Processing of this statement includes loop nesting calculations.

A.2.23 ELSE STATEMENT

Syntax (DEC, F77, SFORT):

```
ELSE
```

SAP does not process beyond the ELSE keyword.

A.2.24 ELSEIF STATEMENT

Syntax (DEC, F77):

```
ELSEIF(e)THEN
```

SAP processing does not include the THEN keyword.

A.2.25 ENCODE STATEMENT

Syntax (DEC):

```
ENCODE(c,f,b[,IOSTAT=ios][,ERR=s]) list
```

The statement label, s, is the only item searched for inside the parentheses.

A.2.26 END STATEMENT

Syntax:

```
END
```

SAP does not process beyond the END keyword. Module statistic collection is complete when this statement is detected.

A.2.27 ENDDO STATEMENT

Syntax (DEC, SFORT):

```
ENDDO
```

SAP does not process beyond the ENDDO keyword.

ORIGINAL PAGE IS
OF POOR QUALITY

A.2.28 ENDFILE STATEMENT

Syntax:

```
ENDFILE u
```

Syntax (also for DEC, F77):

```
ENDFILE (alist)
```

where alist consists of selections from below

```
[UNIT=] u
IOSTAT = ios
ERR = s
```

The statement label, s, is the only item searched for in alist by SAP.

A.2.29 ENDIF STATEMENT

Syntax (DEC, F77, SFORT):

```
ENDIF
```

No processing by SAP beyond the ENDIF keyword. SAP performs some calculations on block IF nesting.

A.2.30 ENTRY STATEMENT

Syntax:

```
ENTRY en [([d[,d]...])]
```

The name en is flagged as an ENTRY name. Each d is flagged as an argument to the module. An * appearing as one of the arguments is ignored.

A.2.31 EQUIVALENCE STATEMENT

Syntax:

```
EQUIVALENCE (nlist)[,(nlist)]...
```

All names appearing in nlist are marked as equivalenced.

ORIGINAL PAGE IS
OF POOR QUALITY

A.2.32 EXTERNAL STATEMENT

Syntax:

```
EXTERNAL proc[,proc]...
```

All procs are marked as EXTERNAL.

A.2.33 FIND STATEMENT

Syntax (IBM):

```
FIND (u'r)
```

Syntax (DEC):

```
FIND (u'r[,IOSTAT=ios][,ERR=s])
```

or

```
FIND ([UNIT=]u,REC=r[,IOSTAT=ios][,ERR=s])
```

The statement label, s, is the only item searched for inside the parentheses.

A.2.34 FORMAT STATEMENT

Syntax:

```
FORMAT fs
```

SAP does not process beyond the FORMAT keyword.

A.2.35 FUNCTION STATEMENT

Syntax:

```
[typ] FUNCTION fun[*n] ([d][,d]...)
```

Each d is flagged as an argument to the module. An * appearing as one of the arguments is ignored. The name fun is used as the module name.

A.2.36 GOTO STATEMENT

Syntax:

```
GOTO s                      unconditional
GOTO (s[,s]...)[,]i         computed
GOTO i[[,](s[,s]...)]       assigned
```

ORIGINAL PAGE IS
OF POOR QUALITY

The index of the computed GOTO is assumed to be a single unsubscripted variable name. The assigned GOTO is not examined beyond index i.

A.2.37 IF STATEMENT

Syntax:

IF(e) s_1, s_2, s_3	arithmetic
IF(e) st	logical

Syntax (DEC, F77):

IF(e) THEN	block

The statement labels of the arithmetic IF are not examined by SAP. The object statement of the logical IF, st, is parsed on a second pass. A logical IF is thus counted as two statements.

A.2.38 .IF STATEMENT

Syntax (SFORT):

.IF(e)	structured

A.2.39 IMPLICIT STATEMENT

Syntax:

```
IMPLICIT typ (a[,a]...)[,typ(a[,a]...)]...
```

Character ranges, a, for type CHARACTER are the only ranges noted by SAP. Any untyped variable name beginning with a character in a CHARACTER range is flagged as a CHARACTER variable name.

A.2.40 INCLUDE STATEMENT

Syntax (DEC):

```
INCLUDE 'filespec'[/[NO]LIST]
```

When the /XP switch is off, there is no statement processing by SAP beyond the INCLUDE keyword. When the /XP switch is on, the source code contained in filespec is processed by

ORIGINAL PAGE IS
OF POOR QUALITY

SAP before proceeding to the following statements. The /LIST or /NOLIST switch is not examined by SAP.

A.2.41 INQUIRE STATEMENT

Syntax (DEC, F77):

```
    INQUIRE (ilist)
```

where ilist consists of selections from the following

```
    [UNIT=]u
    FILE = fin
    IOSTAT = ios
    ERR = s
    EXIST = ex
    OPENED = od
    NUMBER = num
    NAMED = nmd
    NAME = fn
    ACCESS = acc
    SEQUENTIAL = seq
    DIRECT = dir
    FORM = fm
    FORMATTED = fmt
    UNFORMATTED = unf
    RECL = rcl
    NEXTREC = nr
    BLANK = blnk
```

In the DEC dialect of FORTRAN, the following also may be used

```
    CARRIAGECONTROL = cc
    KEYED = kyd
    ORGANIZATION = org
    RECORDTYPE = rty
```

The statement label, s, is the only item in ilist searched for by SAP.

A.2.42 INTEGER STATEMENT

Syntax (F77):

```
INTEGER v[,v]...
```

Syntax (DEC, IBM):

```
INTEGER[*n] v[/clist/][,v[/clist/]...
```

Statement parsing is limited to flagging variable names as array or nonarray. All names are marked as numeric.

A.2.43 INTRINSIC STATEMENT

Syntax (DEC, F77):

```
INTRINSIC fun[,fun]...
```

SAP does not process beyond the INTRINSIC keyword.

A.2.44 LOGICAL STATEMENT

Syntax (F77):

```
LOGICAL v[,v]...
```

Syntax (DEC, IBM):

```
LOGICAL[*n] v[/clist][,v[/clist/]...
```

Statement parsing is limited to flagging variable names as array or nonarray. All names are marked as numeric.

A.2.45 NAMELIST STATEMENT

Syntax (IBM, NPP):

```
NAMELIST /nam/list[,/nam/list]...
```

SAP does not process beyond the NAMELIST keyword.

ORIGINAL PAGE IS
OF POOR QUALITY

A.2.46 OPEN STATEMENT

Syntax (DEC, F77):

```
    OPEN (olist)
```

where olist consists of selections from the following

```
    [UNIT=] u
    IOSTAT = ios
    ERR = s
    FILE = fin
    STATUS = sta
    ACCESS = acc
    FORM =fm
    RECL = rl
    BLANK = blnk
```

In the DEC dialect of FORTRAN, the following also may be used

```
    ASSOCIATEVARIABLE = asv
    BLOCKSIZE = blks
    BUFFERCOUNT = bfr
    CARRIAGECONTROL = cc
    DISP = di
    DISPOSE = dis
    EXTENDSIZE = ext
    INITIALSIZE = ini
    KEY = key
    MAXREC = mrc
    NAME = nam
    NOSPANBLOCKS = nos
    ORGANIZATION = org
    READONLY = rd
    RECORDSIZE = rsz
    RECORDTYPE = rty
    SHARED = shr
    TYPE = typ
    USEROPEN = uop
```

The statement label, s, is the only item in olist searched for by SAP.

A.2.47 PARAMETER STATEMENT

Syntax (F77, DEC):

```
PARAMETER (p=e[,p=e]...)
```

Syntax (also for DEC):

```
PARAMETER p=e[,p=e]...
```

No processing is done by SAP beyond the PARAMETER keyword.

A.2.48 PAUSE STATEMENT

Syntax:

```
PAUSE [n]
```

No processing is done by SAP beyond PAUSE keyword.

A.2.49 PRINT STATEMENT

Syntax:

```
PRINT f[,iolist]
```

Syntax (also DEC, F77):

```
PRINT *[,iolist]
```

A.2.50 PROGRAM STATEMENT

Syntax (DEC, F77):

```
PROGRAM pgm
```

The name pgm is used as the module name. A main program module that does not contain a PROGRAM statement has a default name of MAIN.

A.2.51 READ STATEMENT

Syntax:

```
    READ (clist) [iolist]
or  READ f [,iolist]
```

Syntax (also DEC, IBM):

```
    READ (u'r[,f][,ERR=s]) [iolist]
```

where clist consists of selections from below

```
    [UNIT=] u
    [FMT=] f
    REC = rn
    IOSTAT = ios
    ERR = s1
    END = s2
```

The statement labels, s_1 and s_2, are the only items in clist searched for by SAP.

A.2.52 REAL STATEMENT

Syntax (F77):

```
    REAL v[,v]...
```

Syntax (DEC, IBM):

```
    REAL[*n] v[/clist/][,v[/clist]...
```

Statement parsing is limited to flagging variable names as array or nonarray. All names are marked as numeric.

A.2.53 RETURN STATEMENT

Syntax:

```
    RETURN [e]
```

The expression is not examined. If the RETURN keyword is not the last item in the statement, a RETURN I is counted.

A.2.54 REWIND STATEMENT

Syntax:

```
REWIND u
```

Syntax (also DEC, F77):

```
REWIND [alist]
```

where alist consists of selections from the following

```
[UNIT=] u
IOSTAT = ios
ERR = s
```

The statement label, s, is the only item searched for in alist by SAP.

A.2.55 REWRITE STATEMENT

Syntax (DEC):

```
REWRITE (clist) [iolist]
```

where clist consists of selections from below

```
[UNIT=] u
[FMT=] f
IOSTAT = ios
ERR = s
```

The statement label, s, is the only item searched for in clist by SAP.

A.2.56 SAVE STATEMENT

Syntax (DEC, F77):

```
SAVE [a[,a]...]
```

No processing is done by SAP beyond the SAVE keyword.

A.2.57 STOP STATEMENT

Syntax:

```
STOP [n]
```

No processing is done by SAP beyond the STOP keyword.

A.2.58 SUBROUTINE STATEMENT

Syntax:

```
SUBROUTINE nam [([d[,d]...])]
```

Each d is flagged as an argument to the module. An * appearing as one of the arguments is ignored. The name nam is used as the name of the module.

A.2.59 THEN STATEMENT

This statement is not a valid FORTRAN statement in any dialect acceptable to SAP. It is an artificial construct used at one time to prevent SAP from diagnosing a syntax error when parsing a block IF statement.

A.2.60 TYPE STATEMENT

Syntax (DEC):

```
    TYPE f[,list]
or  TYPE *[,list]
```

A.2.61 UNLOCK STATEMENT

Syntax (DEC):

```
    UNLOCK u
or  UNLOCK (alist)
```

where alist consists of selections from below

```
    [UNIT=] u
    IOSTAT = ios
    ERR = s
```

The statement label, s, is the only item searched for in alist by SAP.

A.2.62 VIRTUAL STATEMENT

Syntax (DEC):

```
    VIRTUAL a(d)[,a(d)]...
```

All names are flagged as array.

A.2.63 WRITE STATEMENT

Syntax:

```
WRITE(clist) [iolist]
```

Syntax (DEC, IBM):

```
WRITE(u'r[,f][,ERR=s]) [iolist]
```

where clist consists of selections from below

```
[UNIT=] u
[FMT=] f
REC = rn
IOSTAT = ios
ERR = s
```

The statement label, s, is the only item searched for in clist by SAP.

APPENDIX B - HALSTEAD'S MEASURES IN SAP

B.1 INTRODUCTION

The measures and counts described by Halstead (Reference 9) are accumulated by SAP as each statement is analyzed. The basic quantities measured during analysis by SAP are as follows:

- n_1 = number of unique or distinct operators appearing
- n_2 = number of unique or distinct operands appearing
- N_1 = total usage of all operators appearing
- N_2 = total usage of all operands appearing
- n_2^* = number of unique input/output parameters to the module

The software metrics calculated from these five basic quantities are described in Section 3.5 and in Reference 9.

This appendix describes how each of the basic quantities is counted. This information is provided to assist researchers in judging how to interpret and apply the results reported by SAP. The interpretation of the basic quantities or of the metrics derived from them is not presented here, but is discussed in Reference 9. Halstead also indicates (Reference 15) how some FORTRAN language structures should be broken down into operators and operands. Halstead's techniques form the basis for the manner in which SAP parses statements to obtain the counts of operators and operands.

The basic quantities are reported in three places in SAP printed output: the complexity paragraph of the module statistic report (Section 2.3.2 and Figure 2-5), the module directory (Section 2.3.1.1 and Figure 2-1), and the project

ORIGINAL PAGE IS
OF POOR QUALITY

summary report (Section 2.3.1.3 and Figure 2-3). In addition, a complete listing of the individual operators and operands detected by SAP is available for each module through the use of the /HL switch (Section 2.3.2 and Figure 2-6).

All five basic quantities are written to the two external SAP files: the SAP data base file (Section 2.3.4) and the ALL.SAP sequential output file (Section 2.3.5).

The following subsections (B.2 through B.4) describe how SAP collects the counts for operators, operands, and input/output parameters, respectively. Most of the tables that appear in these sections contain descriptions of FORTRAN statement syntax. The names and symbols used in these descriptions are taken from the ANSI FORTRAN standards publication (Reference 1). The statements that are not part of the ANSI standard are described as they appear in the VAX FORTRAN manual (Reference 4). Section B.5 presents a sample of source code and a detailed accounting of how the Halstead counts are obtained.

B.2 COUNTING HALSTEAD OPERATORS

SAP counts Halstead operators in four groups: delimiter operators, keyword operators, procedure operators, and transfer operators. The following subsections discuss each type of operator.

B.2.1 DELIMITER OPERATORS

Decomposition of each statement by SAP results in a table of delimiters and tokens for the statement. As portions of particular statements are parsed by SAP, delimiters in the table are tested for membership in the list of delimiter operators (Table B-1). Each occurrence of a delimiter operator in the indicated portion of the particular statements shown in Table B-2 is counted.

ORIGINAL PAGE IS
OF POOR QUALITY

Table B-1. Delimiter Operators

Delimiter Symbol	Definition in Context
//	Character string concatenation
**	Exponentiation
*	Multiplication or list directed input/output format identifier or alternate return specifier
/	Division
+	Addition or positive or floating-point exponent sign
-	Subtraction or negative or floating-point exponent sign
=	Replacement
(	Grouping
,	Separator
&	Alternate return specifier
.NE.	Comparison, not equal
.LT.	Comparison, less than
.LE.	Comparison, less than or equal
.EQ.	Comparison, equal
.GE.	Comparison, greater than or equal
.GT.	Comparison, greater than
.AND.	Logical conjunction
.OR.	Logical inclusive disjunction
.XOR.	Logical nonequivalence
.EQV.	Logical equivalence
.NOT.	Logical negation
.NEQV.	Logical nonequivalence

ORIGINAL PAGE IS
OF POOR QUALITY

At the completion of processing for a module, each delimiter operator with a nonzero count is included in the count of distinct operators appearing (n_1), and the total count of each delimiter operator is added to the count of total usage of operators appearing (N_1).

B.2.2 KEYWORD OPERATORS

The keyword operators are associated with the FORTRAN language structures. The keyword operators are counted each time the particular statement(s) associated with a keyword is encountered. Table B-3 lists the keyword operators as they are labeled on the operator/operand report (Figures 2-6 and B-3), their operation, and the statement types that cause the keyword to be counted.

At the completion of processing for a module, each keyword operator with a nonzero count is included in the count of distinct operators appearing (n_1), and the total count of each keyword operator is added to the count of total usage of operators appearing (N_1).

B.2.3 PROCEDURE OPERATORS

References to external procedures are classified as procedure operators. Procedure operators are counted each time a reference to an external function is detected within a portion of a particular statement. Table B-4 lists the statements and indicates the portion of each statement examined for procedure operators.

At the completion of processing for a module, each procedure operator detected is included in the count of distinct operators appearing (n_1), and the total count of each procedure operator is added to the count of total usage of operators appearing (N_1).

ORIGINAL PAGE IS
OF POOR QUALITY

Table B-2. Statement Types Examined for Delimiter Operators (1 of 2)

Statement Type	Syntax	Portion of Statement Examined for Delimiter Operators
Assignment	v=e	Entire statement
Arithmetic Statement Function Definition	fun([d[,d]...])=e	Entire statement
CALL	CALL sub[([a[,a]...])]	Entire statement following (but not including) open parenthesis preceding the argument list if present
DOWHILE	DO[s[,]]WHILE(e)	Entire statement following and including the open parenthesis preceding the expression
Computed GOTO	GOTO(s[,s]...)[,]i	Entire statement following (but not including) the open parenthesis preceding the statement label list
ELSE	ELSE	No delimiter ever present
THEN	THEN	No delimiter ever present
ELSEIF	ELSEIF(e)THEN	Expression only (not including the enclosing parentheses)
Arithmetic IF	IF(e)s_1,s_2,s_3	Expression only (not including the enclosing parentheses); the statement label list is not examined
Structured IF	.IF(e)	Expression only (not including the enclosing parentheses)

ORIGINAL PAGE IS
OF POOR QUALITY

Table B-2. Statement Types Examined for Delimiter Operators (2 of 2)

Statement Type	Syntax	Portion of Statement Examined for Delimiter Operators
Logical IF	IF(e)st	Expression only (not including the enclosing parentheses); st is not examined as part of the analysis of the logical IF (it is analyzed as a separate statement on a separate pass)
Block IF	IF(e)THEN	Expression only (not including the enclosing parentheses)

ORIGINAL PAGE IS
OF POOR QUALITY

Table B-3. Keyword Operators

Label on Operator/ Operand Report	Operation	Statement Types Causing 1 Count
IF()	Decision	Logical IF
IF(),,	Decision	Arithmetic IF
.IF()	Decision	Structured IF Block IF
ELSEIF	Decision	ELSEIF
ELSE	Alternative	ELSE
DO=,,	Loop Definition	DO
DOWHILE	Loop Definition	DOWHILE
ASSIGNTO	Transfer Selection	ASSIGN
EOS	End of statement	Assignment Arithmetic statement function definition ASSIGN CALL with argument list DO DOWHILE Unconditional GOTO Computed GOTO Assigned GOTO Logical IF Arithmetic IF Structured IF Block IF ELSEIF Input/output statements with END= or ERR= inside parentheses

C-2

ORIGINAL PAGE IS
OF POOR QUALITY

Table B-4. Statement Types Examined for Procedure Operators

Statement Type	Syntax	Portion of Statement Examined for Procedure Operators
Assignment	v=e	Entire statement
Arithmetic Statement Function Definition	fun([d[,d]...])	Entire statement
CALL	CALL sub[([a,...,a])]	sub is counted
DOWHILE	DO[s[,]]WHILE(e)	Expression
ELSEIF	ELSEIF(e)THEN	Expression
ELSE	ELSE	Procedures never present
THEN	THEN	Procedures never present
Arithmetic IF	IF(e)s_1,s_2,s_3	Expression
Structured IF	.IF(e)	Expression
Logical IF	IF(e)st	Expression only; st is not examined as part of the analysis of the logical IF (it is analyzed as a separate statement on a separate pass)
Block IF	IF(e)THEN	Expression

ORIGINAL PAGE IS
OF POOR QUALITY

B.2.4 TRANSFER OPERATORS

The transfer operators are associated with FORTRAN branches. SAP records each transfer operator as a list of the tokens making up the operator. Table B-5 lists the six types of transfer operators and describes the list of tokens extracted from each statement type. The list of tokens associated with each discovered occurrence of a transfer operator is compared with the list of stored tokens for operators of the same type (unconditional GOTOs, assigned GOTOs, computed GOTOs, ERR=, END=, or alternate returns). The transfer operator is counted if it is a recurrence, or it is added to the list with a count of 1 if it has not been previously identified.

A transfer operator must be identical in type of transfer, length of token list, and order of tokens in the list to be counted as a recurrence. Thus, none of the following sample statements would be counted as the same transfer operator:

Statement	Stored Token List
GOTO 100	100
GOTO 200	200
CALL ABC (*100,X,Y,*200)	ABC,100,200
CALL ABC (*200,X,Y,*100)	ABC,200,100
GOTO (100,200),I	100,200,I
GOTO (100,200,300),I	100,200,300,I
READ (5,100,ERR=200)X	200

At the completion of processing for a module, each transfer operator detected is included in the count of distinct operators appearing (n_1), and the count of each transfer operator is added to the count of total usage of operators appearing (N_1).

ORIGINAL PAGE IS
OF POOR QUALITY

Table B-5. Transfer Operators

Statement Type	Syntax	Stored Tokens Used To Identify A Unique Transfer Operator
Alternate Return	CALL sub[([a[,a]...])]	List = sub, a, ...,a, where each argument (a) in the token list is an alternate return specifier label; this operator exists only if at least one argument is an alternate return
Any I/O statement	IO Keyword (...[,END=s]...)	List = s
Any I/O statement	IO Keyword (...[,ERR=s]...)	List = s
Unconditional GOTO	GOTO s	List = s
Computed GOTO	GOTO(s[,s]...)[,]i	List = s,...,s,i, where the index (i) is included in the token list
Assigned GOTO	GOTOi[[,](s[,s]...)]	List = i, where the statement label list is not included in the token list

ORIGINAL PAGE IS
OF POOR QUALITY

B.3 COUNTING HALSTEAD OPERANDS

The result of the decomposition of each statement by SAP is a table of delimiters and tokens. As portions of particular statements are parsed by SAP, each token in the table is flagged and counted as an operand. Each occurrence of a token that is not a function reference (procedure operator) or a FORTRAN statement keyword is counted when it appears in the indicated portion of the particular statements shown in Table B-6. Because of the way in which SAP parses statements, character and Hollerith constants appearing in the indicated statements are not counted as operands.

At the completion of processing for a module, each flagged token is included in the count of distinct operands appearing (n_2), and the total count of each operand's use is added to the count of total usage of operands appearing (N_2).

B.4 COUNTING HALSTEAD INPUT/OUTPUT PARAMETERS

At the completion of module processing, SAP computes the number of input/output parameters to the module from the sum of the following three counts:

- Total number of variable names passed to the module in the SUBROUTINE or FUNCTION statement
- Total number of variable names appearing in ENTRY statements
- Number of variables and arrays appearing in COMMON statements that are also used in the module

These counts appear individually on the module statistics report.

ORIGINAL PAGE IS
OF POOR QUALITY

Table B-6. Statement Types Examined for Operands (1 of 2)

Statement Type	Syntax	Portion of Statement Examined for Operands
Assignment	v=e	Entire statement
Arithmetic Statement Function Definition	fun([d[,d]...])=e	Entire statement
CALL	CALL sub[([a[,a]...])]	Argument list; alternate return specifier labels are included
DO	DO[s]i=e_1,e_2[,e_3] or DO[S[,]]i=e_1,e_2[,e_3]	Entire statements to the right of, and including, i; s is not counted when present
DOWHILE	DO[s[,]]WHILE(e)	Expression; s is not counted when present
Assigned GOTO	GOTOi[[,](s[,s]...)	i is counted; the s's are not counted
Computed GOTO	GOTO(s[,s]...)[,]i	Entire statement to the right of, and including, the first statement label
Unconditional GOTO	GOTO s	s is counted
ELSE	ELSE	Operands never present
THEN	THEN	Operands never present
ELSEIF	ELSEIF(e)THEN	Expression
Arithmetic IF	IF(e)s_1,s_2,s_3	Expression; s's are not counted
Structured IF	.IF(e)	Expression
Logical IF	IF(e)st	Expression only; st is not examined as part of the analysis of the logical IF (it is analyzed as a separate statement on a separate pass)

ORIGINAL PAGE IS
OF POOR QUALITY

Table B-6. Statement Types Examined for Operands (2 of 2)

Statement Type	Syntax	Portion of Statement Examined for Operands
Block IF	IF(e)THEN	Expression
Any I/O statement	IO Keyword(...[,END=s] ...[,ERR=s]...)iolist	Statement labels following END= and/or ERR= in parentheses

ORIGINAL PAGE IS
OF POOR QUALITY

B.5 COUNTING HALSTEAD OPERATORS AND OPERANDS: AN EXAMPLE

This section presents a sample module and an accounting for the resulting Halstead counts. This example should be used in conjunction with the procedures and tables presented in Sections B.2 and B.3 to understand how SAP performs this analysis.

Figure B-1 contains the sample source code. Subroutine TDIST is used throughout this manual whenever a sample report describes an individual module. TDIST is also the last routine in all summary reports and file listings presented. Thus, this figure may also be used to understand other counts produced by SAP.

Figure B-2 shows a line-by-line summary of Halstead operator and operand counts. Column 1 of Figure B-2 indicates a line number from Figure B-1. Columns 2 through 5 indicate the specific operators identified on each line. These columns also contain line totals for each operator type. Column 6 shows the line totals for operators. Columns 7 and 8 present the specific operands and operand line totals, respectively. The last two lines of Figure B-2 show column totals for both unique occurrences and total usage. The totals for columns 6 and 8 correspond to n_1, N_1, n_2, and N_2.

Although Figure B-2 is not a report produced by SAP, Figure B-3 is the SAP report that summarizes the details of Halstead counting. This report is produced for each module when the /HL control switch is set on.

ORIGINAL PAGE IS
OF POOR QUALITY

```
100         SUBROUTINE TDIST (N, X, Y, DIST)
200   C                                                   PASSED
300         INTEGER    N
400         REAL       X(N), Y(N), DIST
500   C                                                   LOCAL
600         INTEGER    I, MSGNUM, K
700         REAL       XL, YL, DX, DY, X2, Y2, R2, R
800         LOGICAL    ERR
900   C                                                   GLOBAL
1000        REAL       SQRT
1100  C                                                   INITIALIZE
1200        XL = 0.0
1300        YL = 0.0
1400        DIST = 0.0
1500  C                                                   FOR ALL POINTS
1600        DO 200 I=1, N
1700          DX = X(I) - XL
1800          X2 = DX*DX
1900          DY = Y(I) - YL
2000          Y2 = DY*DY
2100  C                                                   CALC./CHECK SEPARATION
2200          R2 = X2 + Y2
2300          CALL VERIFY (R2, ERR)
2400  C                                                   OBTAIN SEPARATION
2500          IF (ERR) THEN
2600            K = I - 1
2700            WRITE (6, 100, ERR=300) K, I
2800    100     FORMAT (1X, 'ERROR, POINTS ', I3, ' AND ', I3,
2900       *            ' TOO CLOSE')
3000            R = 0.0
3100          ELSE
3200            R = SQRT (R2)
3300          END IF
3400  C                                                   ACCUMULATE
3500          DIST = DIST + R
3600          XL = X(I)
3700          YL = Y(I)
3800    200 CONTINUE
3900  C                                                   NORMAL RETURN
4000        RETURN
4100  C                                                   ERROR WRITING MESSAGE
4200    300 CONTINUE
4300        MSGNUM = 27
4400        CALL ERRMSG (MSGNUM, *400)
4500        RETURN
4600  C                                                   UNABLE TO WRITE ANY
4700  C                                                   MESSAGES, ABORT RUN
4800    400 CONTINUE
4900        CALL ABORT
5000  C
5100        END
```

Figure B-1. Sample Source Code

ORIGINAL PAGE IS
OF POOR QUALITY

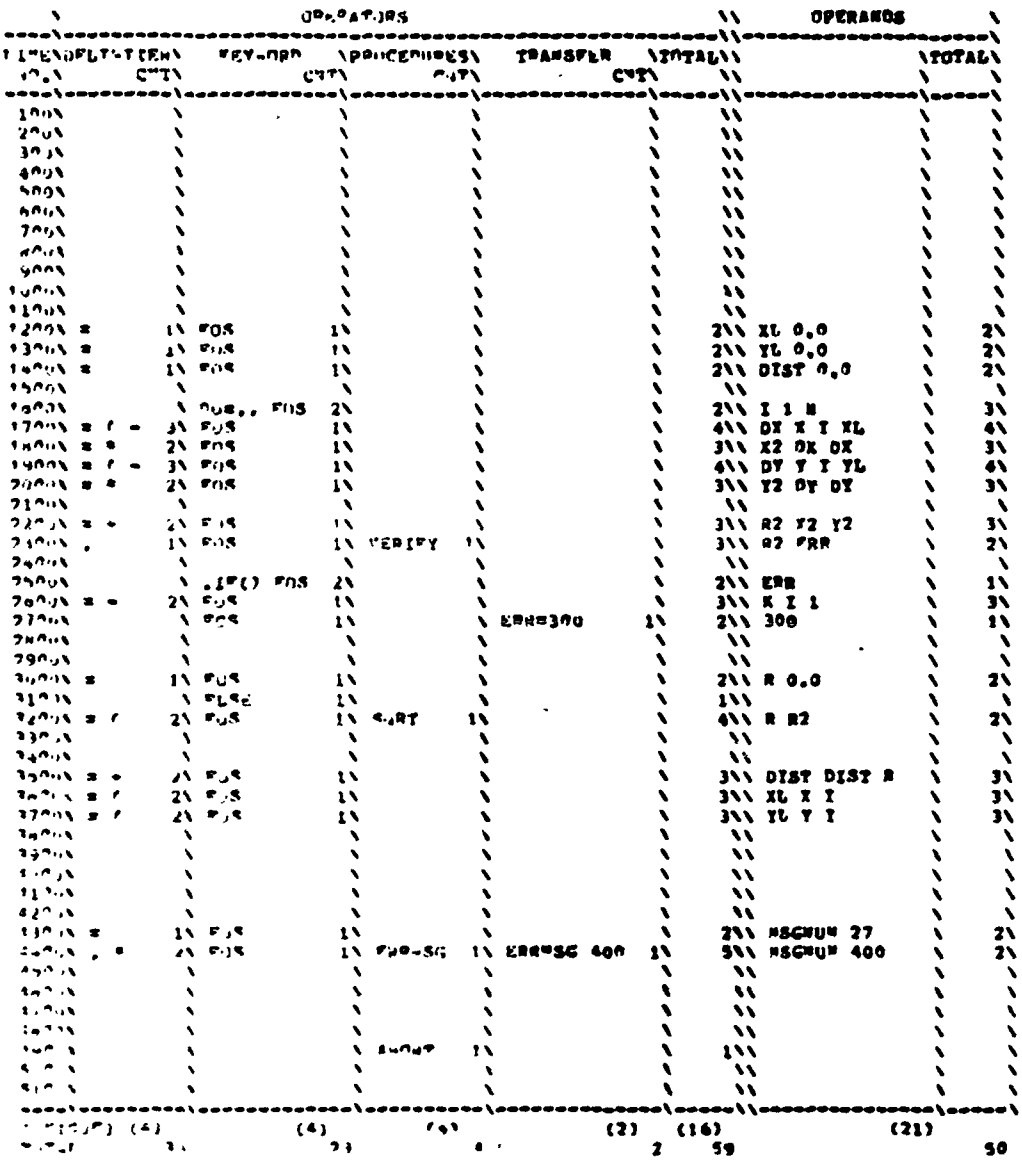

Figure B-2. Line-By-Line Summary of Halstead Counts

```
11:12:46 19-AUG-82  SOURCE ANALYZER PROGRAM V2   MODULE STATISTICS SUMMARY FILE          LUN  7    DIST

SAMPLE.FOR/NL:/OB/SE:/EX

                                    HALSTEAD OPERATORS

DELIMETERS
          0  //       0  **       3  *        0  /        2  +        3  -
         15  =        5  (        2  ,        0  &        0  .NE.     0  .LT.
          0  .LE.     0  .EQ.     0  .GE.     0  .GT.     0  .AND.    0  .OR.
          0  .XOR.    0  .EQV.    0  .NOT.    0  .NEQV.

KEYWORDS
          0  IF()          0  IF(),,       1  .IF()        0  ELSE IF      1  ELSE         1  DO=,,
          0  DOWHILE       0  ASSIGNTO    20  END

PROCEDURES
          1  VERIFY        1  SORT         1  ERRMSG       1  ABORT

TRANSFERS
          1  FMR=      300
          1  ALT,RET,  ERRMSG 400

                                    HALSTEAD OPERANDS

          2 I          6 T          1 K          1 N          3 8          2 X          2 Y          1 27
          3 N2         2 X2         2 Y2         4 0.0        1 300        1 400        3 DX         3 DY
          4 XI         1 YL         2 ERR        3 DIST       2 MSGNUM
```

Figure B-3. SAP Operand/Operator Summary Report

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX C - McCABE'S MEASURE IN SAP

C.1 CYCLOMATIC COMPLEXITY

McCabe's measure (also referred to as the cyclomatic complexity) is described in Reference 10 where it is proposed as an indicator of computer program complexity. The quantity measured is the number of linearly independent paths in a program. The measure is developed from graph theory in which the cyclomatic number V(G) is the maximum number of linearly independent circuits in a strongly connected graph. The expression for the cyclomatic number is

$$V(G) = e - n + 2p \qquad \text{(C-1)}$$

where e = number of edges
n = number of vertices
p = number of connected components (usually one)

McCabe shows that for a program with unique entry and exit nodes, the cyclomatic number for the program control graph is

$$V(G) = d + 1 \qquad \text{(C-2)}$$

where d = number of decisions in the program

The information in this appendix provides a basis that may be used by researchers to judge how and when to use the value of the cyclomatic complexity calculated by SAP. The interpretation of the cyclomatic number as a measure of program complexity is not presented here, but is discussed in Reference 10. McCabe also indicates how some of the FORTRAN language structures contribute to the cyclomatic complexity. McCabe's techniques form the basis for the manner in which SAP calculates the cyclomatic complexity as presented in Section C.3.

ORIGINAL PAGE IS
OF POOR QUALITY

C.2 SAP REPORTS OF THE CYCLOMATIC COMPLEXITY

The cyclomatic complexity is reported in four places in SAP printed output; the complexity paragraph of the module statistics report (Section 2.3.2 and Figure 2-5), the module directory (Section 2.3.1.1 and Figure 2-1), the project summary report (Section 2.3.1.3 and Figure 2-3), and the global correlation report (Section 2.3.1.3 and Figure 2-4). (It should be noted that the correlation report does not list the cyclomatic complexities for the modules, but instead shows the correlation of the cyclomatic complexity with other source code measures.)

The cyclomatic complexity is not written to either of the SAP external data files. However, a related number, the count of decisions, is written to both the SAP data base file (Section 2.3.4) and the sequential output file ALL.SAP (Section 2.3.5). The cyclomatic complexity can be computed from the number of decisions by using Equation (C-2).

C.3 CALCULATION OF THE CYCLOMATIC COMPLEXITY

SAP calculates the cyclomatic complexity from Equation (C-2). The count of decisions in a source code module is obtained by SAP in two stages: (1) during source code parsing, individual contributions to the decision count by various statement types and constructs are accumulated, and (2) the contributions are combined after module parsing is completed. The number of decisions is calculated as

$$d = d(cg) + d(if) + d(do) + d(op) \qquad (C\text{-}3)$$

where d = number of decisions in the module
$d(cg)$ = contribution from computed GOTO statements
$d(if)$ = contribution from IF statements

ORIGINAL PAGE IS
OF POOR QUALITY

d(do) = contribution from looping statements

d(op) = adjustment due to compound decisions (Section C.3.4)

Each contribution is discussed below.

C.3.1 DECISION COUNT FROM COMPUTED GOTO STATEMENTS

The computed GOTO statement is treated as a CASE structure. The number of decisions in an individual computed GOTO statement is calculated from the number of statement labels in the statement label list. It is assumed that the computed GOTO index expression points to one of the statement labels. When this is so, the count of decisions is one less than the number of statement labels in the statement label list. The following computed GOTO would be counted as three decisions:

```
GOTO (100, 200, 300, 400), I
```

SAP obtains the count of decisions by starting after the open parenthesis and counting tokens (including the index expression). The count is then decreased by 2 (not 1) because the index expression is assumed to be a single unsubscripted variable name.

C.3.2 DECISION COUNT FROM IF STATEMENTS

The contribution to the decision count from IF statements is calculated by summing the occurrences of individual logical IF statements, structured IF statements, and block IF statements. Twice the count of individual arithmetic IF statements is added to the above sum.

C.3.3 DECISION COUNT FROM LOOPING STATEMENTS

The contribution to the decision count from looping statements is calculated by summing the occurrences of individual DO and DOWHILE statements.

ORIGINAL PAGE IS
OF POOR QUALITY

C.3.4 DECISION COUNT ADJUSTMENT

An adjustment is made to the decision count because of compound decisions. As shown in Figure C-1, a single block IF statement with a compound expression (logical expressions using the .AND., .OR., .XOR., .EQV., or .NEQV. operators) can be expressed as two or more block IF statments with simple logical expressions. A compound expression using an .AND. or .OR. has one "hidden" decision that is not counted by counting the occurrences of block IF statements. A compound expression using an .XOR., .EQV., or .NEQV. has two hidden decisions.

The counts of logical operator usage are obtained from examination of assignment statements, CALL, DOWHILE, computed GOTO, ENDIF, ELSE, THEN, arithmetic IF, logical IF, structured IF, block IF, and ELSEIF statements. The use counts of .AND., .OR., and .XOR. are summed and used as the adjustment.

ORIGINAL PAGE IS
OF POOR QUALITY

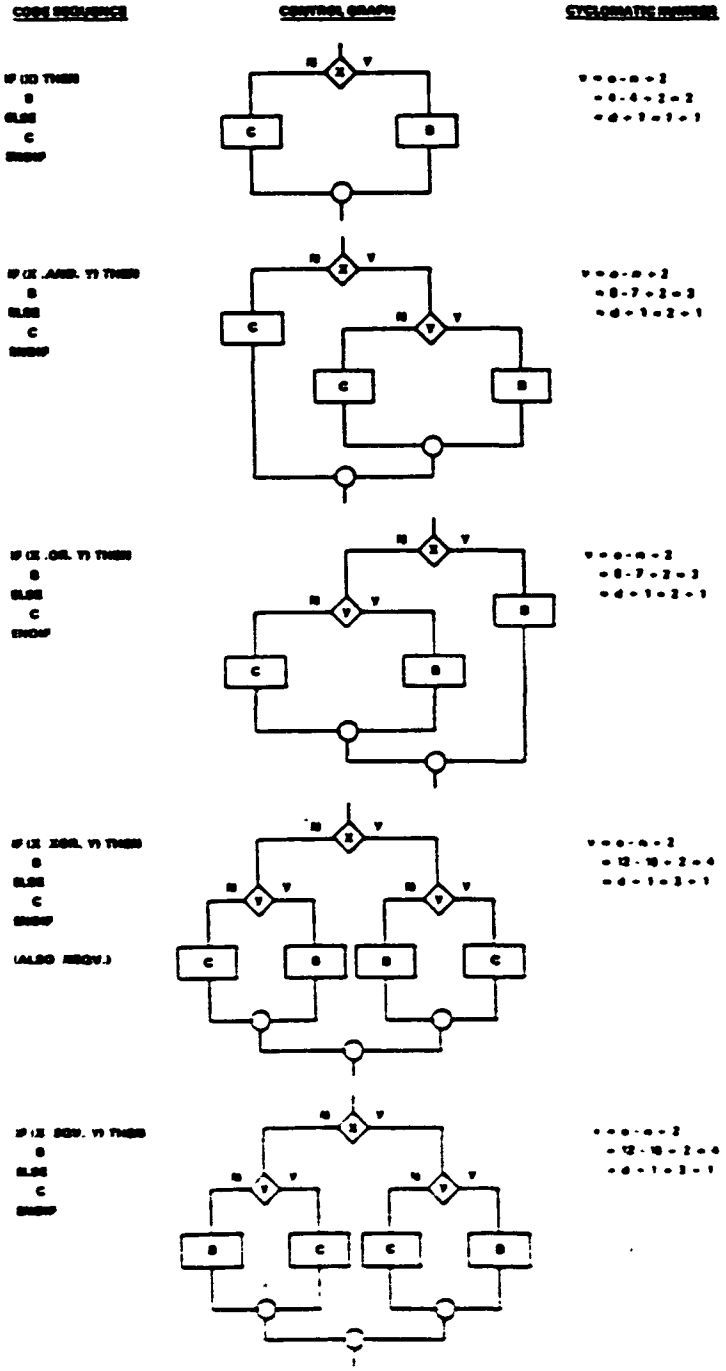

Figure C-1. Control Graphs and Cyclomatic Numbers for Compound Decisions

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX D - USER COMPLEXITY TECHNIQUES

D.1 INTRODUCTION

The SAP user has several options when considering how to compute individualized complexity measures based on the statistics gathered during module processing by SAP. Two options are discussed in this appendix: the SAP statistical weights file and user "stubs" UCPLX1 and UCPLX2. Other options, such as the design of programs to access the SAP sequential or data base files or even the direct modification of code within SAP, are available to the user but their descriptions are beyond the scope of this document.

D.2 USER'S STATISTICAL WEIGHTS FILE

The use of the statistical weights file is briefly discussed in Section 3.4. This file is used to compute a complexity composed of a selected set of SAP statistics in a linear combination. The weighted complexity is displayed in both the module directory (Section 2.3.1.1, Figure 2-1) and the module statistics pages (Section 2.3.2, Figure 2-5). The following example is used to demonstrate how a statistical weights file is designed.

In the example, the user wishes to compute, as a source code measure, the difference between the number of paths into and out of a module. The statistical weights file shown in Figure D-1 can be used to compute such a measure. This figure is referred to in the following explanation.

The first record in the file clears all of the statistical weights to 0. The format for this and all records in the file is 2I5, F6.1.

ORIGINAL PAGE IS
OF POOR QUALITY

1	256	0.0	Initialization
149	149	1.0	Count of END statements
150	150	1.0	Count of ENTRY statements
126	126	-1.0	Count of BLOCK DATA statements
173	173	-1.0	Count of RETURN statements
177	177	-1.0	Count of STOP statements

Figure D-1. Sample User Statistical Weights File

One way to calculate the sample measure is to use the counts of statement types described in Tables 3-7 and 3-8. The number of paths into the module might be calculated by summing the counts of SUBROUTINE, FUNCTION, and PROGRAM statements (statistical weight indexes 178, 155, and 170, respectively from column 8 of Table 3-8) and then adding the count of ENTRY statements (statistical weight index 150). However, this would leave uncounted the implied entrance at the start of a main routine having no PROGRAM statement. Instead, because each module type, except BLOCKDATA, always has at least one path into the module in addition to its ENTRY statements, the design shown in Figure D-1 uses the fact that each module also has exactly one END statement. Thus, the number of paths into the module is calculated from the number of END and ENTRY statements (statistical weight indexes 149 and 150) minus the number of BLOCKDATA statements (statistical weight index 126). Record numbers 2 through 4 in Figure D-1 thus account for the number of paths into a module.

The number of paths out of a module is the sum of the counts of RETURN and STOP statements (statistical weight indexes 173 and 177). These counts are subtracted from the number of paths into the module according to record numbers 5 and 6 of Figure D-1.

ORIGINAL PAGE IS
OF POOR QUALITY

The user statistical weights file may be used whenever the source code measure can be expressed as a weighted sum of the statistics shown in Tables 3-1 through 3-14.

D.3 USER COMPLEXITY STUBS

The user may wish to compute a complexity measure that is not a weighted sum of selected statistics as described in Section D.2. This option is provided to the user through the use of two modules (stubs) currently called by SAP immediately after completing processing of a module. The two modules, UCPLX1 and UCPLX2, may be replaced with user-written routines. The supplied versions of these routines do not perform any calculations.

The user may have access to all COMMON block variables in SAP through the use of the INCLUDE statement. These variables and INCLUDE files are described in Reference 12. The results of any user calculation in these routines are passed to SAP through one argument to each routine. These arguments are output only and are of type REAL*4. The arguments are printed in the module directory (Section 2.3.1.1, Figure 2-1) with format F5.1.

Reference 12 contains information on how the SAP task image is linked. When the user has written either UCPLX1 or UCPLX2 and compiled the module, the object module should replace the supplied module and the SAP task image should be relinked.

To reduce the chance of introducing error into other SAP processes, the user should take care not to alter any variable in COMMON or to perform any input or output to SAP files (Reference 12).

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX E - SAP ERROR MESSAGES

SAP informs the user of abnormal conditions during execution by writing messages to the user's listing file. The errors detected by SAP are, in general, those encountered while opening or reading an external file, syntax errors in the source code encountered while parsing a statement, and conditions that cause an internal table to be exceeded.

Most SAP error messages appear in the following format:

*****routine type***** - msg

where routine = name of the SAP routine that detects the abnormal condition
type = ERROR or WARNING
msg = explanatory text

In general, a message of type ERROR indicates a condition that either causes SAP to stop processing for the module or the entire file, or causes the reported statistics to be misleading. A message of type WARNING indicates a condition that affects only the parsing of a single statement.

Each message originating from SAP is presented below. The message is presented and is followed by an explanation of the probable cause of the error. The messages are arranged according to the alphabetical order of the originating routine.

Message:

***** ADDPOT ERROR ***** - NOT ENOUGH ROOM TO ADD ITEM TO NODE, MAX=nnnn

Explanation: The transfer list table is not large enough to contain all transfer operators in the module.

Originating Subroutine: ADDPOT

ORIGINAL PAGE IS
OF POOR QUALITY

Message:

--INVALID SWITCH--aaa

Explanation: The control switch (aaa) specified by the user could not be identified.

Originating Subroutine: CINPUT

Message:

*****-INVALID FILE-*****a...a

Explanation: The file name (a...a), supplied by the user in response to the SAP> prompt, could not be opened.

Originating Subroutine: CINPUT

Message:

ENCOUNTERED ERROR NUMBER nnnn

Explanation: The FORTRAN error number (nnnn) resulted from attempting to open the file specified as a data base.

Originating Subroutine: DEFINE

Message:

*****ERROR OPENING DATA BASE FILE*****

Explanation: An attempt to open the file ALL.SAP resulted in an error.

Originating Subroutine: DEFSEL

Message:

GARBAGE COLLECTION BEING ATTEMPTED!!

Explanation: The symbol table is not large enough to contain all of the symbols in a module without first removing the deleted symbols.

Originating Subroutine: GARCOL

Message:

*****GLINE ERROR***** - READ ERROR ON LUN nn

Explanation: An error occurred while reading the source code input file.

Originating Subroutine: GLINE

Message:

*****GLINE ERROR***** - LINE LENGTH nnnnn.GT.100

Explanation: A source input line was found that exceeds the maximum allowed length.

Originating Subroutine: GLINE

ORIGINAL PAGE IS
OF POOR QUALITY

Message:

*****HOPTR3 WARNING***** - NO ROOM LEFT FOR SUBR/ENTRY/FUNC aaaaaaaa. IGNORED

Explanation: The procedure operator table is not large enough to contain all procedure operators in the module.

Originating Subroutine: HOPTR3

Message:

*****HSCAN WARNING***** - HOLLERITH FIELD LONGER THAN LINE

Explanation: SAP cannot process a Hollerith field that is continued onto a continuation card.

Originating Subroutine: HSCAN

Message:

INCLUD TERMINATED DUE TO TOO MANY LEVELS

Explanation: SAP cannot expand INCLUDE cards beyond a depth of four.

Originating Subroutine: INCLUD

Message:

*****INVALID FILE***** a...a

The file name (a...a), supplied by the user in response to the SAP> prompt, could not be opened.

Originating Subroutine: INCLUD

Message:

OPEN ERROR - a...a

Explanation: An indirect file (a...a), supplied by the user in response to the SAP> prompt, could not be opened.

Originating Subroutine: INPUT

Message:

FILE READ ERROR

Explanation: An error occurred while reading a command line from an indirect file.

Originating Subroutine: INPUT

ORIGINAL PAGE IS
OF POOR QUALITY

Message:

*****INTGR4 WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: A token that is used in a context in which a statement label is expected could not be converted to a binary integer value.

Originating Subroutine: INTGR4

Message:

*****LOADK ERROR***** - OPEN ERROR ON KEYWORD FILE

Explanation: An error occurred while opening the keyword file.

Originating Subroutine: LOADK

Message:

*****LOADK ERROR***** - READ ERROR ON KEYWORD FILE

Explanation: An error occurred while reading the keyword file.

Originating Subroutine: LOADK

Message:

*****LOADK ERROR***** - ERROR ON WEIGHTS FILE

Explanation: An error occurred while opening or reading either the default or user-specified weights file.

Originating Subroutine: LOADK

Message:

*****LOOKP ERROR-ILLEGAL SYMBOL TABLE ADDRESS*****

Explanation: The symbol table pointer passed to LOOKP does not point to the portion of the table in use.

Originating Subroutine: LOOKP

Message:

*****NEWPOT ERROR***** - NOT ENOUGH ROOM FOR A NEW NODE IN THE TRANSFER LIST, MAX=nnnn

Explanation: The transfer table is not large enough to contain all transfer operators in the module.

Originating Subroutine: NEWPOT

ORIGINAL PAGE IS
OF POOR QUALITY

Message:

*****POKEP ERROR - ADDRESS OR LENGTH BAD IN SYMBOL TABLE STOW OPERATION

Explanation: The symbol table pointer passed to POKEP does not point within the bounds of the symbol table, or the symbol will not fit into the symbol table at the indicated position.

Originating Subroutine: POKEP

Message:

*****POKES ERROR***** - SYMBOL TABLE FULL

Explanation: The symbol table is not large enough to contain all the symbols identified in the module.

Originating Subroutine: POKES

Message:

*****PRCALL WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: Three conditions can cause this error: the CALL keyword is the only item in the statement, the subroutine name was used previously as other than a subroutine name, or the first item (if any) following the subroutine name is not an open parenthesis.

Originating Subroutine: PRCALL

Message:

PRDOS: STACK POINTER ERROR

Explanation: The stack used to store DO loop target labels is not large enough to store all the target labels in the module.

Originating Subroutine: PRDOS

Message:

PRDOS: ERRONEOUS DOWHILE STATEMENT

Explanation: Keyword WHILE in a DOWHILE statement is not followed by an open parenthesis, or DO loop control variable WHILE is not followed by an equal sign.

Originating Subroutine: PRDOS

ORIGINAL PAGE IS
OF POOR QUALITY

Message:

*****PRGOTO WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: A constant follows the GOTO keyword and is not the last item in the statement, or the delimiter following the GOTO keyword is not an open parenthesis.

Originating Subroutine: PRGOTO

Message:

PRIFS: STRUCTURED .IF ERROR

Explanation: The closing parenthesis that follows the logical expression is not the last item in the statement.

Originating Subroutine: PRIFS

Message:

PRIFS: ERROR ON ELSE-IF-THEN

Explanation: The last item in the ELSEIF statement is not keyword THEN.

Originating Subroutine: PRIFS

Message:

SYNTAX ERROR IN IMPLICIT STATEMENT

Explanation: Four conditions can cause this error message: (1) the token following an asterisk indicating the length of a variable type cannot be converted to a binary integer; (2) the variable type is not recognized; (3) a delimiter is found in the alphabetic range specifier that is not an open parenthesis, a comma, a minus sign, or a close parenthesis; or (4) a delimiter is found where a variable type is expected.

Originating Subroutine: PRIMPL

Message:

*****PRIO WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: The open parenthesis immediately following the I/O keyword is not matched with a closing parenthesis.

Originating Subroutine: PRIO

Message:

*****PRSPEC WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: Three conditions may cause this error message: (1) a COMMON block name is not enclosed by slashes, (2) a delimiter other than a comma is used in a DIMENSION

ORIGINAL PAGE IS
OF POOR QUALITY

statement, or (3) there are unmatched parentheses in an array declaration.

Originating Subroutine: PRSPEC

Message:

ERROR POPPING STACK ON ENDDO

Explanation: An ENDDO statement is encountered that matches a corresponding DO or DOWHILE statement with a label reference.

Originating Subroutine: PRSTRC

Message:

*****PRSUBS WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: An ENTRY keyword is not followed by an entry point name.

Originating Subroutine: PRSUBS

Message:

*****PRTOKE WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: A token that is not a constant starts with a character other than a letter, a dollar sign, or a percent sign.

Originating Subroutine: PRTOKE

Message:

*****READER WARNING***** - INPUT CARD LENGTH GREATER THAN 1440

Explanation: A statement containing more than 1440 characters has been encountered.

Originating Subroutine: READER

Message:

****USRWTS: ERROR READING USER WEIGHTS FILE

Explanation: An error occurred while reading the user's weights file.

Originating Subroutine: USRWTS

Message:

****USRWTS: ERROR OPENING DEFAULT WEIGHTS FILE

Explanation: An error occurred while opening the weights file containing the default weights.

Originating Subroutine: USRWTS

ORIGINAL PAGE 13
OF POOR QUALITY

Message:

****USRWTS: ERROR READING DEFAULT WEIGHTS FILE

Explanation: An error occurred while reading the default weights file.

Originating Subroutine: USRWTS

Message:

*****WRTDB ERROR***** - I/O ERROR IN READING DATA BASE HEADER

Explanation: An error occurred while reading the first record in the data base file to obtain the maximum number of records allowed in the file.

Originating Subroutine: WRTDB

Message:

*****WRTDB WARNING***** - NO ROOM LEFT IN DATA BASE FOR ADDITIONAL DATA. NOTHING WRITTEN

Explanation: The data base file contains the maximum number of records specified by the user and an attempt was made to write more data to the file.

Originating Subroutine: WRTDB

ORIGINAL PAGE IS
OF POOR QUALITY

REFERENCES

1. American National Standards Institute, ANSI X3.9-1978, American National Standard Programming Language FORTRAN, April 1978

2. Digital Equipment Corporation, AA-1855D-TC, PDP-11 FORTRAN Language Reference Manual, December 1979

3. --, AA-1884C-TC, FORTRAN IV-PLUS User's Guide, December 1979

4. --, AA-D034B-TE, VAX-11 FORTRAN Language Reference Manual, April 1980

5. --, AA-D035B-TE, VAX-11 FORTRAN User's Guide, April 1980

6. International Business Machines Corporation, SC28-6852, IBM OS FORTRAN IV (H Extended) Compiler Programmer's Guide, November 1974

7. --, GC28-6515, IBM System /360 and System /370 FORTRAN IV Language, May 1974

8. Software Engineering Laboratory, SEL-77-003, Structured FORTRAN Preprocessor (SPORT), B. Chu, D. Wilson, and R. Beard, September 1977

9. M. Halstead, Elements of Software Science. New York: Elsevier Publishing Co., 1977

10. T. J. McCabe, "A Complexity Measure," IEEE Transactions on Software Engineering, December 1976, vol. 2, no. 4, pp. 308-320

11. International Business Machines Corporation, GC28-6586, OS Utilities, April 1973

12. Software Engineering Laboratory, SEL-82-002, FORTRAN Static Source Code Analyzer Program (SAP) System Description, W. Decker, W. Taylor, et al., August 1982

13. --, SEL-81-002, Software Engineering Laboratory (SEL) Data Base Organization and User's Guide, D. Wyckoff, G. Page, F. McGarry, et al., September 1981

14. J. Stroud, "The Fine Structure of Psychological Time," Annals of New York Academy of Sciences, 1966

ORIGINAL PAGE 13
OF POOR QUALITY

15. N. Bulut, M. Halstead, and R. Bayer, "Experimental Validation of A Structural Property of FORTRAN Algorithms," Proceedings of the ACM Annual Conference, November 1974, pp. 207-211

ORIGINAL PAGE IS
OF POOR QUALITY

BIBLIOGRAPHY OF SEL LITERATURE

The technical papers, memorandums, and documents listed in this bibliography are organized into two groups. The first group is composed of documents issued by the Software Engineering Laboratory (SEL) during its research and development activities. The second group includes materials that were published elsewhere but pertain to SEL activities.

SEL-Originated Documents

SEL-76-001, Proceedings From the First Summer Software Engineering Workshop, August 1976

SEL-77-001, The Software Engineering Laboratory, V. R. Basili, M. V. Zelkowitz, F. E. McGarry, et al., May 1977

SEL-77-002, Proceedings From the Second Summer Software Engineering Workshop, September 1977

SEL-77-003, Structured FORTRAN Preprocessor (SFORT), B. Chu, D. S. Wilson, and R. Beard, September 1977

SEL-77-004, GSFC NAVPAK Design Specifications Languages Study, P. A. Scheffer and C. E. Velez, October 1977

SEL-78-001, FORTRAN Static Source Code Analyzer (SAP) Design and Module Descriptions, E. M. O'Neill, S. R. Waligora, and C. E. Goorevich, January 1978

SEL-78-002, FORTRAN Static Source Code Analyzer (SAP) User's Guide, E. M. O'Neill, S. R. Waligora, and C. E. Goorevich, February 1978

SEL-78-102, FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 1), W. J. Decker and W. A. Taylor, September 1982

SEL-78-003, Evaluation of Draper NAVPAK Software Design, K. Tasaki and F. E. McGarry, June 1978

This document superseded by revised document.

ORIGINAL PAGE IS
OF POOR QUALITY

SEL-78-004, Structured FORTRAN Preprocessor (SFORT) PDP-11/70 User's Guide, D. S. Wilson, B. Chu, and G. Page, September 1978

SEL-78-005, Proceedings From the Third Summer Software Engineering Workshop, September 1978

SEL-78-006, GSFC Software Engineering Research Requirements Analysis Study, P. A. Scheffer, November 1978

SEL-78-007, Applicability of the Rayleigh Curve to the SEL Environment, T. E. Mapp, December 1978

SEL-79-001, SIMPL-D Data Base Reference Manual, M. V. Zelkowitz, July 1979

SEL-79-002, The Software Engineering Laboratory: Relationship Equations, K. Freburger and V. R. Basili, May 1979

SEL-79-003, Common Software Module Repository (CSMR) System Description and User's Guide, C. E. Goorevich, S. R. Waligora, and A. L. Green, August 1979

SEL-79-004, Evaluation of the Caine, Farber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Design Environment, C. E. Goorevich, A. L. Green, and P. E. McGarry, September 1979

SEL-79-005, Proceedings From the Fourth Summer Software Engineering Workshop, November 1979

SEL-80-001, Functional Requirements/Specifications for Code 580 Configuration Analysis Tool (CAT), F. K. Banks, C. E. Goorevich, and A. L. Green, February 1980

SEL-80-002, Multi-Level Expression Design Language-Requirement Level (MEDL-R) System Evaluation, W. J. Decker, C. E. Goorevich, and A. L. Green, May 1980

SEL-80-003, Multimission Modular Spacecraft Ground Support Software System (MMS/GSSS) State-of-the-Art Computer Systems/Compatibility Study, T. Welden, M. McClellan, P. Liebertz, et al., May 1980

SEL-80-004, System Description and User's Guide for Code 580 Configuration Analysis Tool (CAT), F. K. Banks, W. J. Decker, J. G. Garranan, et al., October 1980

SEL-80-005, A Study of the Musa Reliability Model, A. M. Miller, November 1980

ORIGINAL PAGE IS
OF POOR QUALITY

SEL-80-006, Proceedings From the Fifth Annual Software Engineering Workshop, November 1980

SEL-80-007, An Appraisal of Selected Cost/Resource Estimation Models for Software Systems, J. F. Cook and F. E. McGarry, December 1980

SEL-81-001, Guide to Data Collection, V. E. Church, D. N. Card, F. E. McGarry, et al., September 1981

SEL-81-002, Software Engineering Laboratory (SEL) Data Base Organization and User's Guide, D. C. Wyckoff, G. Page, F. E. McGarry, et al., September 1981

SEL-81-003, Software Engineering Laboratory (SEL) Data Base Maintenance System (DBAM) User's Guide and System Description, D. N. Card, D. C. Wyckoff, G. Page, et al., September 1981

†SEL-81-004, The Software Engineering Laboratory, D. N. Card, F. E. McGarry, G. Page, et al., September 1981

SEL-81-104, The Software Engineering Laboratory, D. N. Card, F. E. McGarry, G. Page, et al., February 1982

†SEL-81-005, Standard Approach to Software Development, V. E. Church, F. E. McGarry, G. Page, et al., September 1981

SEL-81-105, Recommended Approach to Software Development, S. Eslinger, F. E. McGarry, V. E. Church, et al., May 1982

SEL-81-006, Software Engineering Laboratory (SEL) Document Library (DOCLIB) System Description and User's Guide, W. Taylor and W. J. Decker, December 1981

†SEL-81-007, Software Engineering Laboratory (SEL) Compendium of Tools, W. J. Decker, E. J. Smith, A. L. Green, et al., February 1981

SEL-81-107, Software Engineering Laboratory (SEL) Compendium of Tools, W. J. Decker, E. J. Smith, W. A. Taylor, et al., February 1982

SEL-81-008, Cost and Reliability Estimation Models (CAREM) User's Guide, J. F. Cook and E. Edwards, February 1981

†This document superseded by revised document.

ORIGINAL PAGE IS
OF POOR QUALITY

SEL-81-009, Software Engineering Laboratory Programmer Workbench Phase 1 Evaluation, W. J. Decker, A. L. Green, and F. E. McGarry, March 1981

SEL-81-010, Performance and Evaluation of an Independent Software Verification and Integration Process, G. Page and F. E. McGarry, May 1981

SEL-81-011, Evaluating Software Development by Analysis of Change Data, D. M. Weiss, November 1981

SEL-81-012, The Rayleigh Curve As a Model for Effort Distribution Over the Life of Medium Scale Software Systems, G. O. Picasso, December 1981 (also published as University of Maryland Technical Report TR-1186, July 1982)

SEL-81-013, Proceedings From the Sixth Annual Software Engineering Workshop, December 1981

SEL-81-014, Automated Collection of Software Engineering Data in the Software Engineering Laboratory (SEL), A. L. Green, W. J. Decker, and F. E. McGarry, September 1981

SEL-82-001, Evaluation of Management Measures of Software Development, D. N. Card, G. Page, and F. E. McGarry, September 1982, vols. 1 and 2

SEL-82-002, FORTRAN Static Source Code Analyzer Program (SAP) System Description, W. Taylor and W. Decker, August 1982

SEL-82-003, Software Engineering Laboratory (SEL) Data Base Reporting Software User's Guide and System Description, P. Lo and S. Eslinger, September 1982

SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982

SEL-Related Literature

Anderson, L., "SEL Library Software User's Guide," Computer Sciences-Technicolor Associates, Technical Memorandum, June 1980

ORIGINAL PAGE IS
OF POOR QUALITY

†Bailey, J. W., and V. R. Basili, "A Meta-Model for Software Development Resource Expenditures," Proceedings of the Fifth International Conference on Software Engineering. New York: Computer Societies Press, 1981

Banks, F. K., "Configuration Analysis Tool (CAT) Design," Computer Sciences Corporation, Technical Memorandum, March 1980

††Basili, V. R., "The Software Engineering Laboratory: Objectives," Proceedings of the Fifteenth Annual Conference on Computer Personnel Research, August 1977

††Basili, V. R., "Models and Metrics for Software Management and Engineering," ASME Advances in Computer Technology, January 1980, vol. 1

Basili, V. R., "SEL Relationships for Programming Measurement and Estimation," University of Maryland, Technical Memorandum, October 1980

Basili, V. R., Tutorial on Models and Metrics for Software Management and Engineering. New York: Computer Societies Press, 1980 (also designated SEL-80-008)

††Basili, V. R., and J. Beane, "Can the Parr Curve Help with Manpower Distribution and Resource Estimation Problems?", Journal of Systems and Software, February 1981, vol. 2, no. 1

††Basili, V. R., and K. Freburger, "Programming Measurement and Estimation in the Software Engineering Laboratory," Journal of Systems and Software, February 1981, vol. 2, no. 1

††Basili, V. R., and T. Phillips, "Evaluating and Comparing Software Metrics in the Software Engineering Laboratory," Proceedings of the ACM SIGMETRICS Symposium/Workshop: Quality Metrics, March 1981

Basili, V. R., and T. Phillips, "Validating Metrics on Project Data," University of Maryland, Technical Memorandum, December 1981

††This article also appears in SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982.

ORIGINAL PAGE IS
OF POOR QUALITY

Basili, V. R., and R. Reiter, "Evaluating Automatable Measures for Software Development," Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity and Cost, October 1979

Basili, V. R., and M. V. Zelkowitz, "Designing a Software Measurement Experiment," Proceedings of the Software Life Cycle Management Workshop, September 1977

††Basili, V. R., and M. V. Zelkowitz, "Operation of the Software Engineering Laboratory," Proceedings of the Second Software Life Cycle Management Workshop, August 1978

††Basili, V. R., and M. V. Zelkowitz, "Measuring Software Development Characteristics in the Local Environment," Computers and Structures, August 1978, vol. 10

Basili, V. R., and M. V. Zelkowitz, "Analyzing Medium Scale Software Development," Proceedings of the Third International Conference on Software Engineering. New York: Computer Societies Press, 1978

Card, D. N., "Early Estimation of Resource Expenditures and Program Size," Computer Sciences Corporation, Technical Memorandum, June 1982

††Chen, E., and M. V. Zelkowitz, "Use of Cluster Analysis to Evaluate Software Engineering Methodologies," Proceedings of the Fifth International Conference on Software Engineering. New York: Computer Societies Press, 1981

Church, V. E., "User's Guides for SEL PDP-11/70 Programs," Computer Sciences Corporation, Technical Memorandum, March 1980

Freburger, K., "A Model of the Software Life Cycle" (paper prepared for the University of Maryland, December 1978)

Higher Order Software, Inc., TR-9, A Demonstration of AXES for NAVPAK, M. Hamilton and S. Zeldin, September 1977 (also designated SEL-77-005)

Hislop, G., "Some Tests of Halstead Measures" (paper prepared for the University of Maryland, December 1978)

††This article also appears in SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982.

ORIGINAL PAGE IS
OF POOR QUALITY

Lange, S. F., "A Child's Garden of Complexity Measures" (paper prepared for the University of Maryland, December 1978)

Miller, A. M., "A Survey of Several Reliability Models" (paper prepared for the University of Maryland, December 1978)

National Aeronautics and Space Administration (NASA), NASA Software Research Technology Workshop (proceedings), March 1980

Page, G., "Software Engineering Course Evaluation," Computer Sciences Corporation, Tecnnical Memorandum, December 1977

Parr, F., and D. Weiss, "Concepts Used in the Change Report Form," NASA, Goddard Space Flight Center, Technical Memorandum, May 1978

Perricone, B. T., "Relationships Between Computer Software and Associated Errors: Empirical Investigation" (paper prepared for the University of Maryland, December 1981)

Reiter, R. W., "The Nature, Organization, Measurement, and Management of Software Complexity" (paper prepared for the University of Maryland, December 1976)

Scheffer, P. A., and C. E. Velez, "GSFC NAVPAK Design Higher Order Languages Study: Addendum," Martin Marietta Corporation, Technical Memorandum, September 1977

Turner, C., G. Caron, and G. Brement, "NASA/SEL Data Compendium," Data and Analysis Center for Software, Special Publication, April 1981

Turner, C., and G. Caron, "A Comparison of RADC and NASA/SEL Software Development Data," Data and Analysis Center for Software, Special Puclication, May 1981

Weiss, D. M., "Error and Change Analysis," Naval Research Laboratory, Tecnnical Memorandum, December 1977

Williamson, I. M., "Resource Model Testing and Information," Naval Research Laboratory, Technical Memorandum, July 1979

ORIGINAL PAGE IS
OF POOR QUALITY

††Zelkowitz, M. V., "Resource Estimation for Medium Scale Software Projects," Proceedings of the Twelfth Conference on the Interface of Statistics and Computer Science. New York: Computer Societies Press, 1979

Zelkowitz, M. V., and V. R. Basili, "Operational Aspects of a Software Measurement Facility," Proceedings of the Software Life Cycle Management Workshop, September 1977

††This article also appears in SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982.