HP 64000 Logic Development System

Model 64330 Series High Level Software Analyzers For 68000/68008/68010

HEWLETT PACKARD

KPR #: D200040972 Product: See product list on front cover. 02.01

Keywords: Z203

One-line description:
The analyzer does not work with #include statements in C or PASCAL

Problem:
The high level language analyzer does not work with code that contains #include statements whose files contain executable code. The source code of an included file is not available for display by the analyzer software as it is now written. Included files which contain only type declarations (or statements which are non-executable) pose no problems for the analyzer.

Solution:
If it is necessary to debug code which uses #include statements of files which contain executable code, the user (for the purposes of debugging those executable statements) may merge in the file and eliminate the include statement while he is debugging that portion of the code. After debugging that portion of the code, he can replace the code with the original include statement.

HEWLETT PACKARD

5958-8832, Dec. 1987
E1287

Printed in U.S.A.

HEWLETT PACKARD

OPERATING MANUAL

MODEL 64330 SERIES HIGH LEVEL SOFTWARE ANALYZERS FOR 68000/68008/68010

Manual Part No. 64331-90903
E0985

PRINTED: **September 1985**

PRINTING HISTORY

Each new edition of this manual incorporates all material updated since the previous edition. Manual change sheets are issued between editions, allowing you to correct or insert information in the current edition.

The print date changes only when each new edition is published. Minor corrections or additions may be made as the manual is reprinted between editions. Vertical bars in a page margin indicates the location of reprint corrections.

First Printing .... September, 1984 (P.N. 64331-90902)
Second Edition ....September, 1985 (P.N. 64331-90903)

SOFTWARE VERSION NUMBER

Your HP 64000 software is identified with a version number in the form YY.XX. The version number is printed on a label attached to the software media or media envelope. This manual applies to the following:

Model HP 64331A	Version 2.XX
Model HP 64331B	Version 1.XX
Model HP 64334A	Version 2.XX
Model HP 64334B	Version 1.XX
Model HP 64337A	Version 1.XX

Within the software version number, the digit to the left of the decimal point indicates the product feature set. This manual supports all software versions identified with this same digit.

The digits to the right of the decimal point indicate feature subsets. These feature subsets normally have no affect on the manual. However, if you subscribe to the "Software Material Subscription" (SMS), these subset items are covered in the "Software Response Bulletin" (SRB).

SOFTWARE MATERIALS SUBSCRIPTION

Hewlett-Packard offers a Software Materials Subscription (SMS) to provide you with the most timely and comprehensive information concerning your HP 64000 Logic Development System. This service can maximize the productivity of your HP system by ensuring that you have the latest product enhancements, software revisions, and software reference manuals. For a more detailed description of the SMS, refer to chapter 1.

DUPLICATING SOFTWARE

Before using the flexible disc(s) provided with this product, make a work copy. Retain the original disc(s) as the master copy and use the work copy for daily use. The procedure for duplicating the master flexible disc(s) is included in chapter 2 of this manual.

Specific rights to use one copy of the software product(s) are granted for use on a single, standalone development station or a cluster of development stations which boot from a single mass storage device.

Should your master copy become lost or damaged, replacement discs are available through your Hewlett-Packard sales and service office.

TABLE OF CONTENTS

Chapter 1. GENERAL INFORMATION

INTRODUCTION . . . 1-1
MANUAL APPLICABILITY . . . 1-1
USING SOFTWARE ANALYSIS TO INCREASE SOFTWARE ENGINEERING PRODUCTIVITY . . . 1-2
PREVIEWING MEASUREMENT CAPABILITIES . . . 1-2
Trace Measurements . . . 1-2
TRACE DATA_FLOW . . . 1-3
TRACE MODULES . . . 1-4
TRACE STATEMENTS . . . 1-5
TRACE VARIABLES . . . 1-6
Interactive Capabilities . . . 1-7
BREAKPOINTS . . . 1-7
DISPLAY/MODIFY VARIABLES . . . 1-7
Emulation Control . . . 1-7
BREAK . . . 1-7
LOAD . . . 1-7
RESET . . . 1-7
RUN . . . 1-7
SOFTWARE MATERIALS SUBSCRIPTION . . . 1-8
Software Updates . . . 1-8
Reference Manual Updates . . . 1-8
Software Problem Reporting . . . 1-8
Software Release Bulletins . . . 1-8
Software Status Bulletins . . . 1-8
General User Information . . . 1-9

Chapter 2. INSTALLING THE SOFTWARE ANALYZER

INTRODUCTION . . . 2-1
INSTALLATION . . . 2-1
MAKING DUPLICATE COPIES OF FLOPPY DISC SOFTWARE . . . 2-1

Chapter 3. GETTING STARTED

INTRODUCTION . . . 3-1
MAJOR SOFTKEY LEVELS . . . 3-1
PREPARING THE SYSTEM FOR MEASUREMENTS . . . 3-2
Initial Turn On . . . 3-3
Building Database Files . . . 3-3
GENERATING COMP_DB FILES AT COMPILE AND LINK TIME . . . 3-3
GENERATING COMP_DB FILES USING THE GENERATE_DATABASE UTILITY . . . 3-4
Files Required By The Generate_Database Utility . . . 3-5
Executing the Generate_Database Command . . . 3-5
Defining the Software Breakpoint Vector (HP 64243,64244,64245 Emulators Only) . . . 3-5
Loading And Executing A Program In Emulation . . . 3-8

TABLE OF CONTENTS (Cont'd)

Selecting The Emulation Analysis Mode
(HP 64243,64244,64245 Emulators only) 3-8
Accessing The Software Analyzer 3-8
PERFORMING A BASIC TRACE MODULES MEASUREMENT 3-10
Loading And Running A Program 3-10
Defining A Default Path (Optional) 3-12
Specifying The Acquisition Depth (Optional) 3-13
Setting Up The Trace Specification 3-13
Interpreting The Trace Listing 3-14
SAVING THE CONFIGURATION 3-14
RECOMMENDED PROGRAMMING STYLE 3-16

Chapter 4. BUILDING DATABASE FILES

INTRODUCTION 4-1
SYMBOLIC INTERFACE 4-1
COMP_DB FILES 4-1
BUILDING THE DATABASE FILE 4-2
Compiling Files 4-3
COMPILER SYMBOL FILE 4-3
ASSEMBLER SYMBOL FILE 4-3
Linking Files 4-3
Using The Generate_Database (gen_db) Command 4-4
REQUIRED FILES 4-4
GENERATE_DATABASE COMMAND SYNTAX 4-5
GENERATE_DATABASE COMMAND PARAMETERS 4-5
GENERATE_DATABASE COMMAND EXAMPLES 4-5
VERIFYING DATABASE FILES 4-6
USING COMPILER DIRECTIVES 4-7
AMNESIA 4-7
ASMB_SYM 4-7
FIXED_PARAMETERS (C only) 4-7
LINE_NUMBERS 4-7
OPTIMIZE 4-7
FILES WRITTEN IN ASSEMBLY LANGUAGE 4-8

Chapter 5. CONTROLLING THE EMULATOR

INTRODUCTION 5-1
EMULATION INTERFACE 5-1
Emulation Configuration File 5-1
Loading And Running The Program 5-1
Selecting The Emulation Analysis Mode
(HP 64243,64244,64245 Emulators only) 5-2
COMMUNICATION BETWEEN THE SOFTWARE ANALYZER AND EMULATION 5-2
USING THE EMULATION MONITOR 5-3
EMULATION COMMANDS 5-3
Break Command 5-4
Load Command 5-5

TABLE OF CONTENTS (Cont'd)

Reset Command 5-7
Run Command 5-8

Chapter 6. MAKING TRACE MEASUREMENTS

INTRODUCTION 6-1
GENERAL 6-1
THE TRACE FILE 6-1
MAKING SEQUENTIAL TRACE MEASUREMENTS 6-2
MEASUREMENT CONSIDERATIONS 6-2
MEASUREMENT SYSTEM INTERFACE 6-3
SIMULATED I/O 6-3
TRACE DATA_FLOW MEASUREMENT 6-3
Command Syntax 6-4
Parameters 6-5
Examples 6-6
TRACE MODULES MEASUREMENT 6-8
Command Syntax 6-8
Parameters 6-8
Examples 6-9
TRACE STATEMENTS MEASUREMENT 6-11
Command Syntax 6-11
Parameters 6-12
Examples 6-12
TRACE VARIABLES MEASUREMENT 6-14
Command Syntax 6-15
Parameters 6-16
Examples 6-16

Chapter 7. USING INTERACTIVE COMMANDS FOR PROGRAM DEBUGGING

INTRODUCTION 7-1
BREAKPOINTS 7-2
Command Syntax 7-2
Parameters 7-2
Examples 7-3
DISPLAY COMMAND 7-4
Command Syntax 7-4
Parameters 7-4
Examples 7-5
MODIFY COMMAND 7-6
Command Syntax 7-6
Parameters 7-7
Examples 7-8

TABLE OF CONTENTS (Cont'd)

Chapter 8. SELECTING AND FORMATTING THE MEASUREMENT DISPLAY

INTRODUCTION 8-1
VIEWING THE MEASUREMENT DATA 8-1
DISPLAY FIELDS 8-1
Source Field 8-1
Source Path Field 8-2
Symbol Field 8-2
Symbol Path Field 8-2
Value Field 8-2
Status Field 8-2
INTERPRETING THE DISPLAY 8-2
Current Line 8-3
Displaying Pad Bytes 8-4
Displaying Variant Records 8-4
Field and Display Width 8-5
Illegal Values 8-5
Special Values 8-5
Incomplete Access To Variables 8-6
DISPLAY COMMAND 8-6
Command Syntax 8-6
Parameters 8-6
Examples 8-7
COPY COMMAND 8-8
Command Syntax 8-8
Parameters 8-8
Examples 8-9
SHOW COMMAND 8-9
Command Syntax 8-9
STATE NUMBER PROMPT 8-10

Chapter 9. CONFIGURING THE ANALYZER

INTRODUCTION 9-1
GETTING THE MEASUREMENT CONFIGURATION LAST USED 9-1
GETTING A MEASUREMENT CONFIGURATION FROM A CONFIGURATION FILE 9-2
Saving A Measurement Configuration 9-2
Loading A Measurement Configuration 9-3
GETTING A MEASUREMENT CONFIGURATION FROM A COMMAND FILE 9-3

Chapter 10. USING SUPPORT COMMANDS

INTRODUCTION 10-1
SYSTEM SOFTWARE CONVENTIONS 10-1
User Identification 10-1
Directed Syntax 10-1
Entering Numeric Values 10-2

TABLE OF CONTENTS (Cont'd)

Entering Module/Variable Names 10-2
Command Files 10-2
Logging Commands 10-3
File Names 10-3
SOFTKEY UTILITY COMMANDS 10-3
Execute Softkey 10-3
Copy Softkey 10-3
Show Softkey 10-3
End Softkey 10-4
Halt Softkey 10-4
Wait Softkey 10-4
KEYBOARD UTILITY KEYS 10-4
Recall Key 10-5
Tab Key 10-5
Insert Char And Delete Char Keys 10-5
PROMPT SOFTKEYS 10-5

Chapter 11. SYMBOLS AND DATA TYPES

INTRODUCTION 11-1
SYMBOL CLASSIFICATIONS 11-1
Static Symbols 11-1
LOCAL AND GLOBAL VARIABLES 11-1
PROGRAMS, MODULES, PROCEDURES, AND FUNCTIONS 11-1
LABELS 11-2
LINE NUMBERS 11-2
PATHS 11-2
Proc 11-2
File 11-2
Default Path 11-2
Dynamic Symbols 11-3
LOCAL VARIABLES 11-3
REFERENCE PARAMETERS 11-3
VALUE PARAMETERS 11-3
SYMBOLIC DATA TYPES 11-3
Intrinsic Data Types 11-3
Structured Data Types 11-5
ARRAY 11-5
POINTERS 11-5
SET 11-5
RECORD/STRUCTURE 11-5
VARIANT RECORDS/UNIONS 11-5

TABLE OF CONTENTS (Cont'd)

Appendix A. OPERATING SYNTAX DIAGRAMS

Appendix B. STATUS, ERROR, AND SOFTKEY PROMPT MESSAGES

Appendix C. STACK ARCHITECTURE AND MEMORY STRUCTURE

INTRODUCTION C-1
STACK ARCHITECTURE C-1
Pascal Compiler Considerations C-1
C Compiler Considerations C-2

Appendix D. GLOSSARY OF SOFTKEY LABELS

Appendix E. RESOLVING MEASUREMENT PROBLEMS

INTRODUCTION E-1
MEASUREMENT PROBLEMS AND SOLUTIONS E-1
Missing Source Statements E-1
Missing Symbols On The Display E-2
Unexpected Analyzer Execution E-3
Unexpected Emulation Operation E-4
Unexpected Error Or Status Message E-4
Unexpected Source Line E-6
Unexpected Symbols On The Display E-7
Unexpected Termination of Measurement E-7
Unexpected Value On The Display E-7

Appendix F. SPECIFICATIONS

Index

LIST OF ILLUSTRATIONS

1-1. Trace Data_flow Measurement Display 1-3
1-2. Trace Modules Measurement Display 1-4
1-3. Trace Statements Measurement Display 1-5
1-4. Trace Variables Measurements Display 1-6

3-1. Utility Keys Used To Access the Analyzer 3-10
3-2. Listing of Example Pascal Program 3-11
3-3. Software Analyzer Setup Display 3-13
3-4. Trace Modules Measurement Display 3-15

4-1. Software Analyzer Symbolic Interface 4-2
4-2. Generate_Database Command Syntax Diagram 4-5
4-3. Database_check Command Syntax Diagram 4-6

6-1. Setup Trace Data_Flow Syntax Diagram 6-4
6-2. Trace Data_Flow Measurement Display 6-7
6-3. Setup Trace Modules Syntax Diagram 6-8
6-4. Trace Modules Measurement Display 6-10
6-5. Setup Trace Statements Syntax Diagram 6-12
6-6. Trace Statements Measurement Display 6-14
6-7. Setup Trace Variables Syntax Diagram 6-15
6-8. Trace Variables Measurement Display 6-18

7-1. Setup Breakpoints Syntax Diagram 7-2
7-2. Display Variables Syntax Diagram 7-4
7-3. Modify Variables Syntax Diagram 7-6

8-1. Compiler Listing File For Program EXAMPLE 8-3
8-2. Sample Display Showing How Pad Bytes, Variant Records,
and Field Widths Are Displayed 8-4
8-3. Example Display Showing Illegal Values, Special Values,
and Incomplete Access to Values 8-5
8-4. Display Command Syntax Diagram 8-7
8-5. Copy Command Syntax Diagram 8-8
8-6. Show Command Syntax Diagram 8-9

A-1. Software Analyzer Level Syntax Diagram A-2
A-2. Run Syntax Diagram A-3
A-3. Setup Syntax Diagram A-3
A-4. Setup Trace Data_Flow Syntax Diagram A-4
A-5. Setup Trace Modules Syntax Diagram A-4
A-6. Setup Trace Statements Syntax Diagram A-5
A-7. Setup Trace Variables Syntax Diagram A-5
A-8. Setup Breakpoints Syntax Diagram A-6
A-9. Setup Default_Path Syntax Diagram A-6
A-10. Database_check Syntax Diagram A-6
A-11. Display Variables Syntax Diagram A-7
A-12. Modify Variables Syntax Diagram A-7
A-13. Show Command Syntax Diagram A-7
A-14. Display Command Syntax Diagram A-8
A-15. Copy Command Syntax Diagram (Measurement Display) A-8

LIST OF ILLUSTRATIONS (Cont'd)

A-16. Load Command Syntax Diagram A-9
A-17. Configuration Command Syntax Diagram A-9
A-18. Copy Command Syntax Diagram (Setup Display) A-9
A-19. Variable Syntax Diagram A-10
A-20. P_variable Syntax Diagram A-10
A-21. C_variable Syntax Diagram A-10

C-1. Pascal Stack Frame C-2
C-2. C Stack Frame (Fixed Parameters Options On) C-3
C-3. C Stack Frame (Fixed Parameters Options Off) C-4

LIST OF TABLES

11-1. Intrinsic Data Types 11-3

B-1. Status Messages B-1
B-2. Error Messages B-3
B-3. Softkey Prompt Messages B-8

D-1. Software Analyzer Softkey Labels D-1

NOTES

USING THIS MANUAL

The contents of this manual are summarized in the following paragraphs as an aid in locating information.

Chapter 1, General Information, provides an overview of how the software analyzer fits within the environment of the software life cycle. It also gives an overview of the software analyzer's measurement capabilities.

Chapter 2, Installation, describes the system components required to run the analyzer package and the procedures for installing those components. It also describes how to make duplicate copies of the analyzer software for backup.

Chapter 3, Getting Started, provides a description of the major softkey levels, and step-by-step procedures for preparing the system for measurements, configuring a measurement, and saving measurement configurations on disc. This chapter takes the user through the entire measurement process step-by-step, familiarizing the user with the steps required to make a measurement. It also gives guidelines for writing code to achieve the best results from your software analyzer.

Chapter 4, Building Database Files, describes the symbolic interface and how the data base is built when a program is compiled and linked. It also describes how to verify that the database file is correct. Compiler directives and the implications of their use with the software analyzer are also discussed.

Chapter 5, Controlling the Emulator, gives information on loading and running programs with the emulation system. This chapter describes the emulation commands available from within the software analyzer.

Chapter 6, Making Trace Measurements, gives general information about trace measurements and provides comprehensive descriptions of each of the trace measurements. The detailed command syntax for each measurement is given, all parameters used in the commands are defined, and an example of each measurement is shown and explained.

Chapter 7, Using Interactive Commands For Program Debugging, describes how breakpoints and the display and modify commands can be used to interact with the user program to aid in software debugging.

Chapter 8, Selecting and Formatting the Measurement Display, describes the conventions and features of the measurement display, the commands used to select data to be displayed on the CRT, and methods for optimizing the data display.

Chapter 9, Configuring the Analyzer, describes how to both manually and automatically configure the analyzer for measurements.

Chapter 10, Using Support Commands, identifies and describes system software conventions, utility commands, utility keyboard keys, and software prompts.

Chapter 11, Symbols and Data Types, provides information regarding the symbol storage classes and data types that the software analyzer recognizes.

Appendices A through F provide operating syntax diagrams, status and error messages, processor specific information, softkey prompts, a glossary of softkey labels, solutions to measurement problems, and the software analyzer specifications.

An index is provided for quick reference to specific items.

Chapter 1

GENERAL INFORMATION

INTRODUCTION

The HP 64330 Series High Level Software Analyzers are software development tools designed for HP 64000 Development System users developing microprocessor code in Pascal or C languages. The software analyzer provides you with the capability to debug software at the source code level. The analyzer enables users to reference procedures and data structures in a program as they were originally described in the high level source code. The analyzer includes both global and local measurements. Global measurements provide you with an overview of program activity enabling fault isolation to the module level. The local measurements then enable you to pinpoint problems within the faulty module.

MANUAL APPLICABILITY

This manual applies to the following High Level Analyzer products:

HP 64331A for use with HP 64242S Emulation Subsystem (68000)

HP 64331B for use with HP 64243AA/AB Emulation Subsystems (68000)

HP 64334A for use with HP 64249S Emulation Subsystem (68010)

HP 64334B for use with HP 64245AA/AB Emulation Subsystems (68010)

HP 64337A for use with HP 64244AA Emulation Subsystem (68008)

These analyzers require the appropriate emulation subsystem, emulation memory subsystem, HP 64302A Internal Analyzer, and an HP 64815 Pascal or HP 64819 C cross compiler. The software analyzer works with the emulation subsystem. Trace measurements are made using the emulation bus analyzer. The acquired data is processed and displayed in the appropriate format. The code being analyzed must be compiled using the HP 64000 Pascal or C compiler.

NOTE

Unless otherwise specified, explanations and examples in this manual apply to all high level software analyzers listed in the preceding paragraph. Most examples in this manual were generated using the HP 64334A High Level Software Analyzer for 68010 processors.

USING SOFTWARE ANALYSIS TO INCREASE SOFTWARE ENGINEERING PRODUCTIVITY

The software analyzer provides software designers with a powerful tool that can improve their efficiency in both the development and maintenance phases of the software life cycle.

An effective approach to software development is to break an overall software task down into many independent modules with well-defined interfaces. Each module must be implemented, tested, debugged, and integrated with the other modules. The software analyzer with its trace variable and trace statements measurements allows the software designer to view software in a real execution environment at the level it was written, providing insights and understanding of the code that can significantly reduce the time required for test and debug of software modules. Questions such as what is the value of a variable and how was it modified are answered directly.

When integrating modules to form the overall software task, the designer must verify that each module is called in the correct sequence and that parameters passed between modules are correct. The software analyzer's global type measurements were designed for this purpose. The trace modules and trace data_flow measurements provide this global overview of the software task. The trace modules measurement traces procedures and displays what procedures were called, the order they were called, and which source line they were called from. The trace data_flow measurement traces the data being passed to and from procedures using the parameter names defined by the designer. This ability to have a global overview of a software task with data displayed in the same high level constructs defined by the software designer enables fast analysis and identification of problems.

The maintenance phase of the software life cycle can be expensive in terms of time and resources. A large part of the maintenance is involved with fixing bugs and making enhancements to released products. The same analysis needs are present in the maintenance phase that are present in the development phase for new software products. The software analyzer can help reduce costs during the maintenance phase as it does during the development phase by improving the efficiency of software designers.

PREVIEWING MEASUREMENT CAPABILITIES

Trace Measurements

The following paragraphs provide an overview of the trace measurement capabilities of the software analyzer. The use of each measurement and the information displayed in the trace listings are described. Trace measurements are made automatically using the emulation bus analyzer and then the acquired data is processed and displayed on the screen in the selected format. The software analyzer executes four trace measurements. These measurements are; (1) trace data_flow, (2) trace modules, (3) trace statements, and (4) trace variables. These measurements are described in the following paragraphs. For detailed information on how to use the trace measurement commands, refer to chapter 6, Making Trace Measurements. The measurement display formats are described in detail in chapter 8., Selecting and Formatting the Measurement Display.

TRACE DATA_FLOW. The trace data_flow measurement provides an overview of program data flow at the module level. If a module is not executing correctly, this measurement gives a user the capability to correlate input and output data with module execution, isolating the problem to either the input data or the logic of the module itself.

The trace data_flow measurement traces the values of program variables at the entry and/or exit points of specified procedures or functions. Two types of data can be traced; parameters of the procedure or function and variables defined external to the specified procedure or function. Local variables declared within a specified procedure or function cannot be traced in this measurement mode. Local variables do not exist outside of the procedure or function in which they are declared. For each procedure, tracing may be done on entry to the procedure, on exit from the procedure, or both entry and exit. Tracing on entry records the values of the variables on entry to the procedure and tracing on exit records the values on exit from the procedure. Tracing on both entry and exit records both the entry and exit values.

The default trace data_flow measurement display consists of four fields as shown in figure 1-1. These fields are Symbol, Value, Stat, and Source. The symbol field contains the name of the specified procedure or function followed by the names of the traced variables (indented). In the line containing the procedure name, the value field is blank. The stat field indicates whether the line is the entry point or the exit point of the procedure. If the line is the entry point of the procedure, the source field contains the source file statement and line number corresponding to the line from which the procedure is called. In the lines containing the variable symbol, the stat and source fields will be blank and the value field will contain the value of the variable expressed in the notation of its data type (integer, real, etc.).

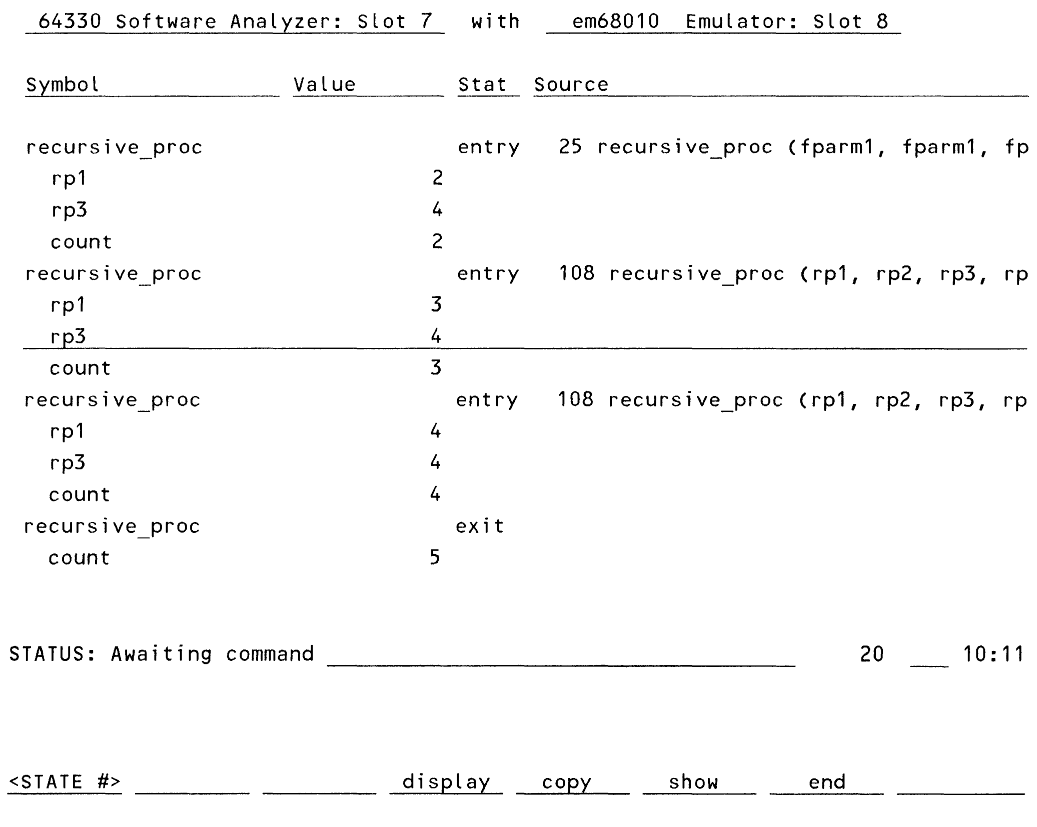

```
  64330 Software Analyzer: Slot 7   with   em68010  Emulator: Slot 8

Symbol                  Value        Stat   Source

recursive_proc                       entry   25 recursive_proc (fparm1, fparm1, fp
  rp1                          2
  rp3                          4
  count                        2
recursive_proc                       entry   108 recursive_proc (rp1, rp2, rp3, rp
  rp1                          3
  rp3                          4
  count                        3
recursive_proc                       entry   108 recursive_proc (rp1, rp2, rp3, rp
  rp1                          4
  rp3                          4
  count                        4
recursive_proc                       exit
  count                        5


STATUS: Awaiting command                                          20       10:11


<STATE #>                   display   copy    show    end
```

Figure 1-1. Trace Data_flow Measurement Display

TRACE MODULES. The trace modules measurement gives you an overview of a program's control flow at the module level. This measurement provides the capability to isolate a problem to a specific module and to provide a history of the module calls leading up to the problem. The trace modules measurement traces the execution sequence of program modules. As procedure and function calls are executed, the module name is listed with indentation used to indicate nesting level. The measurement can be set up to trace all modules in a specific file or group of files or to trace only specific modules in a file or group of files.

The default trace modules measurement display consists of three fields; symbol, status, and source as shown in figure 1-2. The symbol field contains the name of the traced module. Indentation is used to show the nesting level at which the module was called relative to the start of the trace. Figure 1-2 shows how recursive and nested procedures and functions are displayed by the software analyzer. The stat field indicates whether the display line represents the entry point or exit point of the module. If it is the entry point to the module, then the source file statement which called the module and its line number are displayed in the source field.

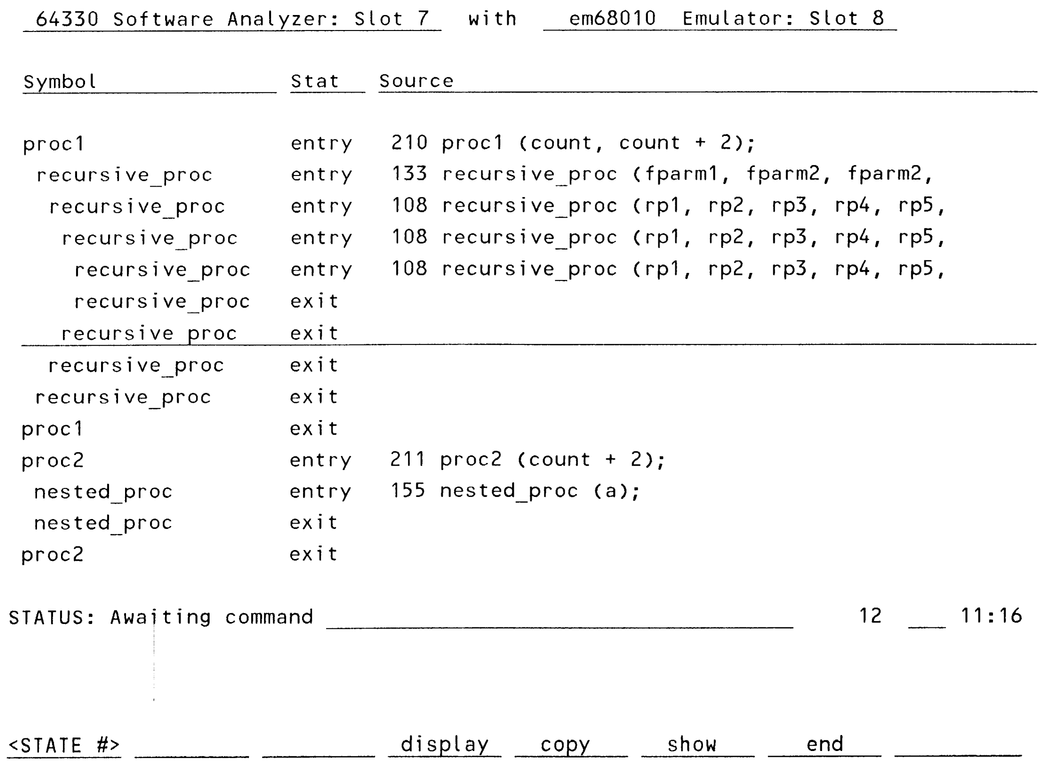
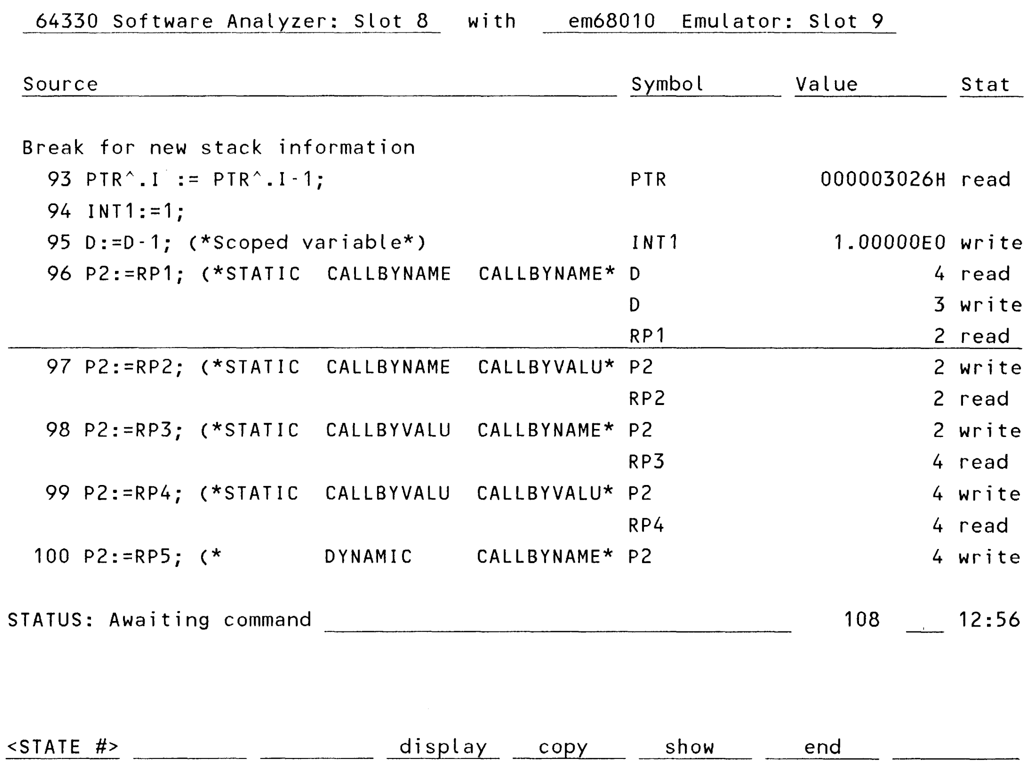

```
 64330 Software Analyzer: Slot 7   with   em68010  Emulator: Slot 8

Symbol               Stat   Source

proc1                entry   210 proc1 (count, count + 2);
 recursive_proc      entry   133 recursive_proc (fparm1, fparm2, fparm2,
  recursive_proc     entry   108 recursive_proc (rp1, rp2, rp3, rp4, rp5,
   recursive_proc    entry   108 recursive_proc (rp1, rp2, rp3, rp4, rp5,
    recursive_proc   entry   108 recursive_proc (rp1, rp2, rp3, rp4, rp5,
    recursive_proc   exit
   recursive_proc    exit
  recursive_proc     exit
 recursive_proc      exit
proc1                exit
proc2                entry   211 proc2 (count + 2);
 nested_proc         entry   155 nested_proc (a);
 nested_proc         exit
proc2                exit

STATUS: Awaiting command                                  12       11:16

<STATE #>                    display    copy    show    end
```

Figure 1-2. Trace Modules Measurement Display

TRACE STATEMENTS. The trace statements measurement gives you a detailed view of a small section of code. Once a problem is isolated to a particular module, the trace statements measurement allows a close inspection of this code to more precisely determine the problem.

The trace statements measurement traces the execution of statements and the value of all variables in each statement in a defined program block. The program block can be defined as a line range, a procedure, or a function. All lines in the specified line range must be contained within a single module. This module may be a procedure, function, or main program block in Pascal, or a function in C.

Figure 1-3 shows a trace statements measurement display. Four fields are displayed in the default trace list; source, symbol, value, and stat. The source field contains the source statement and line number. The symbol field contains the name of the variable accessed by the source statement. If more than one variable is accessed in the source statement, additional variables are displayed on subsequent lines of the display. The value field displays the value of the variable in the data type notation that the variable was declared in the source program. The stat (status) field displays whether the variable access was a memory read or write operation.



```
 64330 Software Analyzer: Slot 8   with   em68010 Emulator: Slot 9

 Source                                        Symbol      Value       Stat

 Break for new stack information
   93 PTR^.I := PTR^.I-1;                      PTR          000003026H read
   94 INT1:=1;
   95 D:=D-1; (*Scoped variable*)              INT1         1.00000E0 write
   96 P2:=RP1; (*STATIC  CALLBYNAME  CALLBYNAME* D                  4 read
                                               D                    3 write
                                               RP1                  2 read
   97 P2:=RP2; (*STATIC  CALLBYNAME  CALLBYVALU* P2                 2 write
                                               RP2                  2 read
   98 P2:=RP3; (*STATIC  CALLBYVALU  CALLBYNAME* P2                 2 write
                                               RP3                  4 read
   99 P2:=RP4; (*STATIC  CALLBYVALU  CALLBYVALU* P2                 4 write
                                               RP4                  4 read
  100 P2:=RP5; (*        DYNAMIC     CALLBYNAME* P2                 4 write

STATUS: Awaiting command                                  108       12:56


<STATE #>                    display   copy   show   end
```

Figure 1-3. Trace Statements Measurement Display

TRACE VARIABLES. Trace variables gives you a history of a set of variables. If a variable is suspected of containing bad data, the variable can be traced to reveal where it is modified and what value it receives. The trace variables measurement enables specific data elements (variables) to be traced and the code that accesses these variables to be located. The variable or variables specified are traced during the execution of the program, and displayed with their values and the source lines from which they were accessed.

The trace variables measurement allows you to trace up to 10 variables. The variables may be in the same file or different files. If the variables are spread out over a large address range in memory, the measurement will take a longer time to fill the trace buffer during execution than if the variables are spread over a small address range. You can qualify the trace variables measurement to trace read only accesses, write only accesses, or both read and write accesses.

A default trace variables measurement display is shown in figure 1-4. The symbol field contains the name of the variable traced. The value field displays the value of the variable. The stat field indicates whether the access to the variable was a read or write operation. The source field shows the source file line number and the source statement.

```
 64330 Software Analyzer: Slot 8    with    em68010  Emulator: Slot 9

 Symbol                  Value            Stat  Source

 Q.I                                 123 write  205 Q.I           :=123;
 Q.REAL_NUMBER                2.00000E0 write  206 Q.REAL_NUMBER :=2;
 Q.FLAG                             TRUE write  206 Q.REAL_NUMBER :=2;
 Q.SETT               [RED,BLUE,WHITE] write  208 Q.SETT        :=[RED,WHITE,BLU
 Q.N                                   0 write  208 Q.SETT        :=[RED,WHITE,BLU
 Q.VARIANT1[RED]                     RED write  210 Q.VARIANT1    :=A;
 Q.VARIANT1[ORANGE]               ORANGE write  210 Q.VARIANT1    :=A;
 Q.VARIANT1[YELLOW]               YELLOW write  210 Q.VARIANT1    :=A;
 Q.VARIANT1[05H?]                    E2H write  210 Q.VARIANT1    :=A;
 Q.CHAR1                             "1" write  211 Q.CHAR1       :="1";
 Q.I                                 123 write  205 Q.I           :=123;
 Q.REAL_NUMBER                2.00000E0 write  206 Q.REAL_NUMBER :=2;
 Q.FLAG                             TRUE write  206 Q.REAL_NUMBER :=2;
 Q.SETT               [RED,BLUE,WHITE] write  208 Q.SETT        :=[RED,WHITE,BLU

STATUS: Awaiting command                                        15     13:01


<STATE #>                      display   copy    show    end
```

Figure 1-4. Trace Variables Measurement Display

Interactive Capabilities

In addition to the trace measurement capabilities, the software analyzer also has commands that allow you to interact with the emulator without exiting the analyzer. These commands are setup breakpoints and display/modify variables.

BREAKPOINTS. The software analyzer provides you with the capability to define breakpoints. This enables you to command the emulation system to perform a "break on execution." The breakpoint may be specified as (1) any valid address of an opcode, (2) any Pascal or C source line containing executable code, or (3) the entry or exit point of a module (procedure or function). Up to 16 software breakpoints can be defined with the setup breakpoints command.

DISPLAY/MODIFY VARIABLES. The display variables command displays the current value of a variable in emulation memory. The variable is specified in the same notation that it is specified in the source file. The modify variables command allows a variable to be modified manually by the user. The variable can be set to a specified value in integer format. The user program must be halted and the emulator running in the emulation monitor before the display or modify commands can be executed. Detailed descriptions of how to use these commands are given in chapter 7, Using Interactive Commands for For Program Debugging.

Emulation Control

The software analyzer includes a subset of the emulator operational commands. The break, load, reset, and run commands enable you to have control over emulation from within the analyzer. These are the four basic commands needed to control a user's program running in emulation memory. The incorporation of the emulator commands simplify the interface between the user and the system by providing the means for the user to control the emulator without exiting the software analyzer.

BREAK. a break causes the processor to be diverted from execution of the user program to the emulation monitor. A break is defined as a transition from execution of a user's program to the emulation monitor.

LOAD. The load command transfers absolute code from the HP 64000 system disc into user RAM or emulation memory. The destination of the absolute code is determined by the memory configuration map which was set up during emulation configuration and the address specified during linking.

RESET. A reset suspends target system operation and reestablishes initial operating parameters, such as reloading control registers. The reset signal is latched when active and is released by the run command.

RUN. If the processor is in a reset state, the run command will cause the reset to be released. If a from address is specified, the processor will be directed to that address. If the processor is running in the emulation monitor, the run command causes the processor to exit into the user program. The program can either be run from (1) the transfer address of the user's program, (2) a specified address, (3) a specified line number in the source code, (4) from the entry point of a specified module, or (5) from the address currently stored in the processor's program counter.

SOFTWARE MATERIALS SUBSCRIPTION

Hewlett-Packard offers a Software Materials Subscription (SMS) to provide you with the most timely and comprehensive information concerning your HP 64000 Logic Development System. This service can maximize the productivity of your HP system by ensuring that you have the latest product enhancements, software revisions, and software reference manuals.

Consult with your local HP Field Representative for a complete list of available software update products (HP 64XXXAU), one-time product updates (HP 64XXXAX), and current prices.

By purchasing SMS, you will obtain the following:

Software Updates
Reference Manual Updates
Software Problem Reporting
Software Release Bulletins
Software Status Bulletins
General User Information

Software Updates

Software Updates may address specific anomalies in HP software or enhance the capability of the HP software in your system.

Reference Manual Updates

Reference manual updates assure that you always have the most recent documentation on a timely basis, and are aware of how to use any new features on the latest software releases.

Software Problem Reporting

Software problem reporting is provided so that you may inform HP of a discrepancy or problem found in the HP 64000 software or documentation.

Software Release Bulletins

Software Release Bulletins document all fixes and enhancements that are incorporated in the latest release of the HP 64000.

Software Status Bulletins

Software status bulletins contain timely information on the reported operational status of HP software and documentation. These bulletins also provide temporary corrections or ways to work around anomalies in HP software which have been located by HP personnel or HP 64000 users. You may reference these bulletins to see of a solution is already documented.

General User Information

General user information is documentation that contains operational tips, programming techniques, application notes, latest listings of software products and reference manuals, and other items of general interest to HP 64000 users.

NOTES

Chapter 2

INSTALLING THE SOFTWARE ANALYZER

INTRODUCTION

A floppy disc containing the analyzer software is required for operation. In addition to the analyzer software, your HP 64000 development station must have a complete emulation system for the processor you are working with, including a Model 64302A Internal Analyzer. Also you must have the HP 64000 Pascal or C compiler for your processor. If your Model 64100A or 64110A development station has a serial number prefix lower than 2309A, you also need a Model 64032A Memory Expansion Module.

INSTALLATION

Install the emulation system in accordance with instructions given in the emulation operating manual. The HP 64032A memory expansion module, if used, may be installed in any unused card slot in the development station card cage.

MAKING DUPLICATE COPIES OF FLOPPY DISC SOFTWARE

Your software analyzer was shipped on two floppy discs. You should make another copy of the floppy discs for your use and protect the original discs that you received from Hewlett-Packard. The following procedure describes how to make duplicate floppy discs so that the original discs may be stored for safekeeping.

To make a duplicate floppy disc, proceed as follows:

1. Remove a new blank floppy disc from its container and label it SOFTWARE ANALYZER. Do not write directly on the floppy disc; this can damage the floppy. Use stick-on labels, if available, or a felt-tip pen.

2. Install the original disc for the software analyzer in disc drive 0 of your 64000 station.

3. Install the new blank SOFTWARE ANALYZER in disc drive 1.

4. From the system monitor softkey level, press the following softkeys in the sequence shown:

   --BACKUP- *floppy* *utilities* RETURN

5. The CRT display will show an explanation of the floppy utilities routines. A floppy disc must be formatted prior to use. Formatting initializes the disc, preparing it to receive information. To format the disc, press the ***format*** softkey and the "1" key, then press the RETURN key.

6. When disc 1 formatting is completed, press the ***duplicate*** softkey and the "0" key, then press the RETURN key. The contents of the software analyzer disc will be duplicated on the blank formatted disc in disc drive 1.

Perform the preceding steps for each of the software analyzer discs. This completes the procedure for making user "SOFTWARE ANALYZER" discs.

Chapter 3

GETTING STARTED

INTRODUCTION

This chapter contains information to help you become familiar with the operation of the software analyzer. You will learn about the first level of analyzer softkeys and how to use them in specifying a measurement. You will learn how to build the database files required by the analyzer. You will also learn to enter measurement specifications in the software analyzer and to gather data as a result of the measurement specifications you set up. In addition, you will learn to save a configuration to a file and reload that configuration at a later time. Guidelines for writing code to achieve the best results from the software analyzer are given at the end of this chapter.

If you have any difficulties or problems when using the software analyzer, see appendix E, Resolving Measurement Problems, for possible solutions.

MAJOR SOFTKEY LEVELS

The software analyzer has a user-friendly interface designed to provide you with easily definable options to examine Pascal and C programs. The user-friendly interface provides a logical structural breakdown and guided syntax softkeys that make definition of measurements easy.

The major softkey levels of the software analyzer are the ***setup***, ***display***, ***modify***, ***show***, ***execute***, ***end***, ***run***, ***break***, ***reset***, ***load***, ***configure***, and ***copy*** softkeys. These softkeys are discussed briefly in the following paragraphs. This brief account of each key allows you to become familiar enough with them to perform the familiarization exercises detailed later in this chapter. A detailed explanation of all the softkeys used in, and under, the major softkey levels is given in later chapters. Syntax diagrams for the major softkey level functions are given in appendix A.

setup	The ***setup*** softkey allows you to specify; (1) the type of trace measurement to be made with the parameters to be traced, (2) the breakpoint condition(s) to stop your program execution, (3) the default path to be used in measurement specifications, i.e., the procedure and/or file information, and (4) the acquisition depth, i.e., the amount of data to be acquired and stored in the analyzer trace file.
display	Pressing the ***display*** softkey and entering a variable name and path (if other than the default path) causes the value of the specified Pascal or C variable to be displayed. The command allows you to specify a variable in a defined program module and absolute file.
modify	The ***modify*** softkey enables you to modify the current value of a variable in memory. The variable can be set to a specific value which must be entered as a simple integer value less than or equal to 32 bits in width.

show	The *show* softkey allows you to switch between the measurement setup display or the measurement data display, providing that valid measurement data is available for display.
execute	The *execute* softkey causes execution of a trace measurement, of software breakpoints, or both.
end	Pressing the *end* softkey one time causes the subsystem to terminate the current measurement session and places the 64000 station back into the measurement system monitor. The software analyzer can be reentered from this level simply by pressing the *sw_anly* softkey. The *end* softkey also saves the current configuration in the default configuration file and updates the emulation command (emul_com) file.
run	The *run* softkey allows you to start execution of the user program in emulation without exiting the software analyzer. When the processor is in a reset state, the *run* command causes the reset to release. When a *from* address is specified, the processor is directed to that address. If the processor is running in emulation monitor, the *run* command causes the processor to exit into the user program.
break	Pressing the *break* softkey causes the processor to be diverted from execution of the user program to the emulation monitor.
reset	The *reset* softkey allows you to suspend target system operation and reestablish initial emulator operating parameters. The reset signal is latched when active and is released by the *run* command.
load	The *load* softkey allows you to transfer absolute code from the 64000 system disc into user RAM or emulation memory. The destination of the absolute code is determined by the memory configuration map which was set up during emulation configuration and the address specified during linking.
configure	The *configure* softkey allows you to either save or load the complete analyzer configuration to or from a file.
copy	The *copy* softkey allows you to copy the measurement setup, measurement data, or the current display to either a file or the printer.
<CMDFILE>	The <CMDFILE> softkey is a prompt informing the user that a command file may be executed at this level to automatically execute software analyzer commands.

PREPARING THE SYSTEM FOR MEASUREMENTS

The information contained in this section is provided to help you become familiar with the basic operation of the software analyzer. You will be lead through the steps required to configure the 64000 system for performing basic software measurements. You will learn how to gain access to the analysis functions and how to setup the analyzer to make a simple trace modules measurement.

Initial Turn On

NOTE

The following procedure assumes that you have installed an emulation system in your development station, you have the floppy disc containing the software analyzer software, and you are in a cluster with a hard disc.

1. Connect operating power to the development station.
2. Turn on the power switch. The associated indicator lamp (on 64110 development stations) will light.
3. Boot in the software analyzer software following the instructions given in the 64000 System Software Manual.
4. When the software has been successfully booted on your system disc, the display will come up with the units connected to the bus identified. The monitor level of softkeys will be displayed on the bottom of the screen.
5. You may, at this time, wish to assign a user identity code to your activity with the station. The software records your userid and assigns any files you may make to your userid. The userid must start with an upper case alphabetic character and is limited to six characters. After the first letter, the other five characters may be alphanumeric. To assign your userid press the ***---ETC---*** softkey twice, press the ***userid*** softkey, type in the userid you have selected, and press the (RETURN) key. If no userid is selected, the default condition is a blank userid.

   ---ETC--- ***---ETC---*** ***userid*** **<USERID>** (RETURN)

Building Database Files

The basis of the software analyzer measurements are the comp_db (compiler database) files. All files to be debugged with the software analyzer must have an associated comp_db file. The comp_db files allow the software analyzer to decode symbols into addresses and the addresses back into symbols. Comp_db files provide information on the symbol types (used for display purposes) and ownership of symbols by functions, procedures, or files. Comp_db files can be generated in two ways; (1) by compiling the source file with option ***comp_sym*** and linking with option ***comp_db***, or (2) by using the generate_database utility (***gen_db***).

GENERATING COMP_DB FILES AT COMPILE AND LINK TIME. The most efficient method of generating comp_db files for source files compiled on your HP 64000 Development Station is to compile the files using the ***comp_sym*** (compiler symbol) option and to link the files using the ***comp_db*** option. The following two steps build the comp_db file required by the software analyzer.

1. Compile all files that you wish to debug using the ***comp_sym*** option. This option specifies the saving of the compiler symbol file, making it available to the linker for use in generating the comp_db file. The command is:

   compile MYFILE ***options*** ... ***comp_sym*** (RETURN)

   The "..." located between ***options*** and ***comp_sym*** indicates that other options may be specified in addition to the ***comp_sym*** option. However, the ***comp_sym*** option must be the last in the list of options.

   The compiler symbol file includes the following information; the processor for which the file was compiled, the language the file was written in, the names, addresses, and data sizes for modules, and the names, types, sizes and locations of variables unique to each module. The compiler symbol file is not automatically saved after each compilation.

   An asmb_sym (assembler symbol) file is created for every file compiled unless the ***nocode*** option is included in the compile command. The contents of the asmb_sym file for a compiled file include local symbol names and relocatable or absolute addresses for those local symbols. Also, the addresses for line numbers are recorded here. In order for the analyzer to execute correctly, the asmb_sym file must be created for each file to be analyzed.

   NOTE

   DO NOT compile the file to be analyzed with the option ***nocode***. This suppresses the creation of an assembly symbol file, a file required for proper operation of the software analyzer.

2. Next, the compiled files must be linked. The command is:

   link LINK_COM_FILE ***options*** ... ***comp_db*** (RETURN)

   The link_sym (linker symbol) file is created during the linking process and contains information about all files included in the link command. Included are global symbol names and their relocated addresses, source names and their relocated addresses, and a list of memory space used by the linked files.

   A database file is created at link time, when ***options comp_db*** is specified, for each file that was compiled with the ***comp_sym*** option. The ***comp_db*** must be the last specified option in the link command, as ***comp_sym*** is in the compile command.

GENERATING COMP_DB FILES USING THE GENERATE_DATABASE UTILITY. The generate_database utility allows you to generate a comp_db file for files developed in a hosted environment using HP 64000 series hosted compilers. The utility also allows you to generate comp_db files for source files developed on an HP 64000 development station, but not compiled with the comp_sym option or linked with the comp_db option. The generate_database utility provides the necessary link for performing high level software analysis in an HP 64000 development station of programs developed in the hosted development environment.

Files Required By The Generate_Database Utility. The generate_database utility requires the following files to be downloaded from the hosted development environment:

Pascal and C source files
Absolute files (.X)
Asmb_sym files (.A)
Link_sym files (.L)

A high level debug files transfer utility is available on the hosted system. This utility transfers all files required by the generate_database utility. See the Hosted Development System User's Guide for detailed information on the transfer utility.

Executing the Generate_Database Command. Executing the following command generates a comp_db file for source file MY_FILE:

generate_database MY_FILE *using* LINK_SYM_FILE

Where LINK_SYM_FILE is a valid link_sym file and MY_FILE is a source file referenced in the link_sym file.

This command first executes pass 1 of the compiler to generate the required comp_sym file. It then uses the asmb_sym, comp_sym, and link_sym files to generate the comp_db file. If a valid comp_sym file exists, then the following command may be executed:

generate_database comp_db MY_FILE *using* LINK_SYM_FILE

This command uses the existing comp_sym file, eliminating compiler pass 1 execution.

Before attempting to use the software analyzer, read chapter 4, Building Database Files. This chapter contains important information on compiling and linking files for analysis.

Defining the Software Breakpoint Vector (HP 64243,64244,64245 Emulators Only)

You must establish a trap vector for software breakpoints for proper operation of the high level software analyzer. If you have not yet defined the software breakpoint vector for using software breakpoints in emulation, you must define the trap vector now, as detailed in the following steps:

1. Edit the emulation monitor program (file Mon_680NN:HP) and find the following code segment (NN = 00, 08, or 10 depending on which emulator you are using):

```
*******************************************************************
*    TRAP VECTORS ( SOFTWARE BREAKPOINT VECTORS ) CHOOSE ONE
*    -----------------------------------------------------
*  The TRAP vector you should choose for the software breakpoint
*  depends on the trap # chosen in the configuration question.
*******************************************************************
* ORG 080H
*    DC,L SWKB_ENTRY     ;TRAP #0
* ORG 084H
*    DC,L SWKB_ENTRY     ;TRAP #1
* ORG 088H
*    DC,L SWKB_ENTRY     ;TRAP #2
* ORG 08CH
*    DC,L SWKB_ENTRY     ;TRAP #3
* ORG 090H
*    DC,L SWKB_ENTRY     ;TRAP #4
* ORG 094H
*    DC,L SWKB_ENTRY     ;TRAP #5
* ORG 098H
*    DC,L SWKB_ENTRY     ;TRAP #6
* ORG 09CH
*    DC,L SWKB_ENTRY     ;TRAP #7
* ORG 0A0H
*    DC,L SWKB_ENTRY     ;TRAP #8
* ORG 0A4H
*    DC,L SWKB_ENTRY     ;TRAP #9
* ORG 0A8H
*    DC,L SWKB_ENTRY     ;TRAP #10
* ORG 0ACH
*    DC,L SWKB_ENTRY     ;TRAP #11
* ORG 0B0H
*    DC,L SWKB_ENTRY     ;TRAP #12
* ORG 0B4H
*    DC,L SWKB_ENTRY     ;TRAP #13
* ORG 0B8H
*    DC,L SWKB_ENTRY     ;TRAP #14
* ORG 0BCH
*    DC,L SWKB_ENTRY     ;TRAP #15
```

2. Remove comment delimiters from the two lines associated with the trap vector you wish to use. (In this example, trap vector 15 is used.)

```
*********************************************************************
*    TRAP VECTORS ( SOFTWARE BREAKPOINT VECTORS ) CHOOSE ONE
*    ------------------------------------------------------
*  The TRAP vector you should choose for the software breakpoint
*  depends on the trap # chosen in the configuration question.
*********************************************************************
* ORG 080H
*    DC,L SWKB_ENTRY     ;TRAP #0
* ORG 084H
*    DC,L SWKB_ENTRY     ;TRAP #1
* ORG 088H
*    DC,L SWKB_ENTRY     ;TRAP #2
* ORG 08CH
*    DC,L SWKB_ENTRY     ;TRAP #3
* ORG 090H
*    DC,L SWKB_ENTRY     ;TRAP #4
* ORG 094H
*    DC,L SWKB_ENTRY     ;TRAP #5
* ORG 098H
*    DC,L SWKB_ENTRY     ;TRAP #6
* ORG 09CH
*    DC,L SWKB_ENTRY     ;TRAP #7
* ORG 0A0H
*    DC,L SWKB_ENTRY     ;TRAP #8
* ORG 0A4H
*    DC,L SWKB_ENTRY     ;TRAP #9
* ORG 0A8H
*    DC,L SWKB_ENTRY     ;TRAP #10
* ORG 0ACH
*    DC,L SWKB_ENTRY     ;TRAP #11
* ORG 0B0H
*    DC,L SWKB_ENTRY     ;TRAP #12
* ORG 0B4H
*    DC,L SWKB_ENTRY     ;TRAP #13
* ORG 0B8H
*    DC,L SWKB_ENTRY     ;TRAP #14
  ORG 0BCH
     DC,L SWKB_ENTRY     ;TRAP #15
```

3. End the edit session.

4. Re-assemble the emulation monitor and relink your program using the ***comp_db*** option.

5. Enter the emulator and modify the configuration. Cycle through the emulation configuration questions until the question "Re-configure emulator pod?" is displayed.

6. Answer "yes" and cycle through the question until the question "Trap number for software break (0..0FH) ?" is displayed.

7. Answer the question with the trap number you selected in the emulation monitor edit session (15 or 0FH in this example).

8. End the emulation configuration session.

Loading And Executing A Program In Emulation

When the 64000 station is turned on, the softkey label line is displayed on the screen and contains the *meas_sys* softkey label. When you press the *meas_sys* softkey and the RETURN key, the *sw_anly* softkey will appear on the softkey label line, along with the name(s) of any emulation system in the development station. This is the measurement system level of softkeys. From here, you will need to enter the emulation system so that you can create an emulation command file which you will need to use the software analyzer. Refer to the emulator operating manual for detailed instructions on creating emulation command files.

Selecting The Emulation Analysis Mode (HP 64243,64244,64245 Emulators only)

The emulator analysis mode must be set to *bus_cycle_data* in order to use the software analyzer. From within the emulation subsystem, execute the command *modify analysis_mode_to bus_cycle_data*. If the emulation analysis mode is not set to *bus_cycle_data*, the error message "**Incorrect analysis bus mode for this analyzer**" is displayed on the status line when you attempt to access the software analyzer.

Accessing The Software Analyzer

NOTE

When an intermodule bus measurement is in progress with other analyzers, the software analyzer cannot be used. The IMB measurement must be halted first.

After you have generated an emulation command file in an emulation session, you are ready to access the software analyzer. Leave the emulation system running and press the *end* softkey and the RETURN key. This will bring you out to the measurement system level of softkeys. You can now access the software analyzer by pressing the *sw_anly* softkey, entering the name of the emulation command file, and pressing RETURN.

sw_anly **<EMUL_COM_FILE>** RETURN

If you omit the emulation command file in the command line, the software analyzer will prompt you for the file;

Emulation command file?

After entering the name of the emulation command file you generated in the emulation session and pressing RETURN, you will access the software analyzer, ready to start your analysis session.

NOTE

On first accessing the software analyzer, you must specify an emulation command file. During the analysis session, you can save the measurement setup in a configuration file. On subsequent uses of the software analyzer, you can specify either a configuration file or an emulation command file in the *sw_anly* command. On ending a measurement session, your last measurement setup is stored in the default configuration file identified with your station HP-IB address and userid.

Figure 3-1 shows the utility softkeys used to gain access to the software analyzer and how to end out of the analyzer and return to the system monitor level of softkeys. Pressing the ***end*** softkey followed by RETURN once will return you to the measurement system level of software. The software analyzer will retain its current measurement setup. To go to the system monitor level of software press the ***end*** again. It is now possible to perform operations at this level (edit, copy, etc.). To reuse the analyzer, and still retain the current measurement setup, press the ***meas_sys*** softkey and ***continue*** softkeys, then the RETURN key. This brings you to the measurement system level of software. Now press the *sw_anly* softkey and then the RETURN key. You are now back in the software analyzer with the current measurement setup retained.

NOTE

Pressing the *sw_anly* softkey when the measurement system is entered with the continue option restores the last measurement setup used in the software analyzer if that session was terminated using the ***end*** command. The emulator hardware will be in the same state it was left in provided that: (1) the 64000 station has not been turned off, (2) the emulator has not been modified during an emulation session, or (3) ***opt_test*** has not been executed.

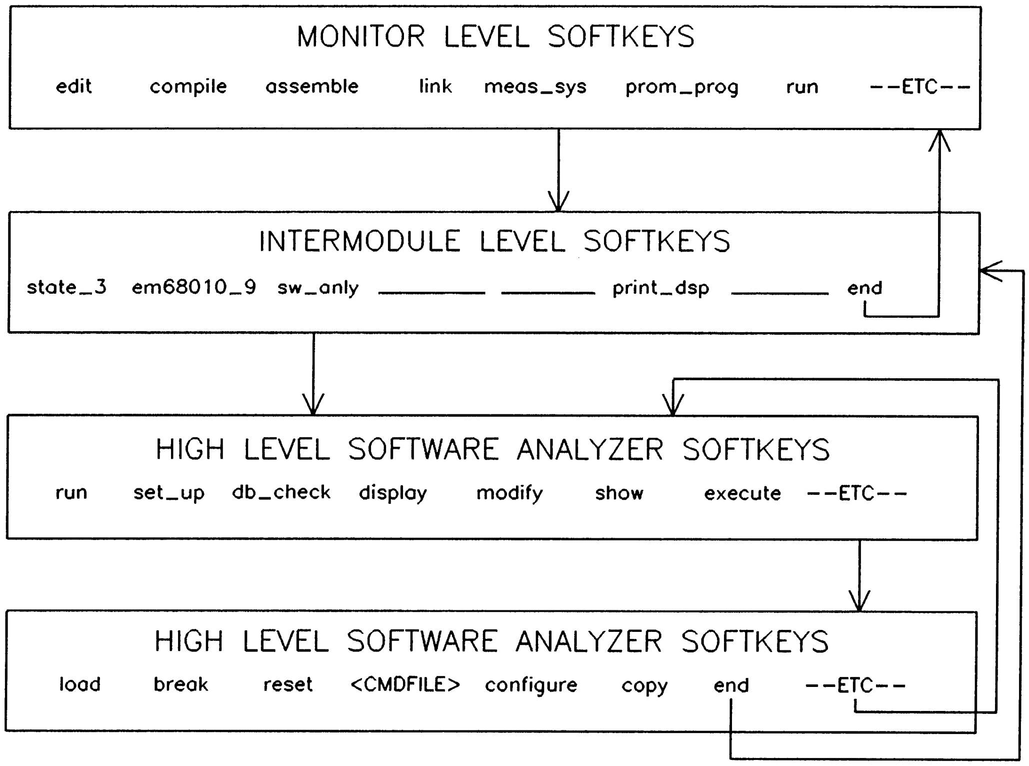

MONITOR LEVEL SOFTKEYS
edit compile assemble link meas_sys prom_prog run --ETC--
INTERMODULE LEVEL SOFTKEYS
state_3 em68010_9 sw_only print_dsp end
HIGH LEVEL SOFTWARE ANALYZER SOFTKEYS
run set_up db_check display modify show execute --ETC--
HIGH LEVEL SOFTWARE ANALYZER SOFTKEYS
load break reset <CMDFILE> configure copy end --ETC--

Figure 3-1. Utility Keys Used To Access the Analyzer

PERFORMING A BASIC TRACE MODULES MEASUREMENT

The following measurement example is intended to familiarize you with the software analyzer as well as show a meaningful measurement on a program written in Pascal. Figure 3-2 shows an outline of the program used in the following example. The program listing shows the procedure declarations and entry and exit points of the procedures traced in the example measurement.

Loading And Running A Program

The first step in preparing to make a measurement is to load the absolute file we wish to analyze into emulation memory. This source file must have been compiled using the *comp_sym* option and the absolute file linked using the *comp_db* option as explained previously. These two options create the symbolic data base required by the software analyzer to interpret and display measurement data. The command is executed by pressing the *load* softkey, typing in the absolute file name, and pressing the RETURN key.

load MYFILE RETURN

The next step is running the program in emulation. Entering the command

run at_execution from transfer_address RETURN causes the user program MYFILE to begin running from its starting address when a measurement is executed. The ***at_execution*** parameter is included here because we wish to ensure that we trace all modules executed from the beginning of the program. Leaving out the ***at_execution*** parameter causes the user program to begin running immediately.

```
 1 00000000  1  "68000"
 6 00000000  1  $WARN+$
 7 00000000  1  $EXTENSIONS ON$
 8 00000000  1  PROGRAM TESTP;
 ...

13 00000000  1  TYPE
14 00000000  1  INT            = SIGNED_16;
15 00000000  1  PTR            =^INT;
16 00000000  1  SCALAR_TYPE    =(BLACK,BROWN,RED,ORANGE,YELLOW,GREEN,BLUE,VIOLET,GREY,WHITE);
17 00000000  1  DAY_OF_WEEK    =(SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY);
18 00000000  1  SUBRANGE_TYPE  =RED..YELLOW;
19 00000000  1  SET_TYPE       =SET OF SCALAR_TYPE;
20 00000000  1  ARRAY_TYPE0    =ARRAY[DAY_OF_WEEK] OF SCALAR_TYPE;
21 00000000  1  ARRAY_TYPE1    =ARRAY[SUBRANGE_TYPE] OF SCALAR_TYPE;
22 00000000  1  ARRAY_TYPE2    =ARRAY[-3..-1] OF BYTE;
23 00000000  1  ARRAY_TYPE3    =ARRAY[0..1] OF ARRAY_TYPE2;
24 00000000  1  REC_TYPE_PTR   =^REC_TYPE;
25 00000000  1  REC_TYPE       =RECORD
26 00000000  2    I            :SIGNED_32;
27 00000000  2    REAL_NUMBER  :REAL;
28 00000000  2    CHAR1        :CHAR;
29 00000000  2    FLAG         :BOOLEAN;
30 00000000  2    SETT         :SET_TYPE;
31 00000000  2    NEXT_REC     :REC_TYPE_PTR;
32 00000000  2
33 00000000  2    CASE N       :BYTE OF
34 00000000  2         1:      (VARIANT1  :ARRAY_TYPE1);
35 00000000  2         2:      (VARIANT2  :ARRAY_TYPE2);
36 00000000  2         3:      (VARIANT3  :ARRAY_TYPE3);
37 00000000  1    END;
 ...

42 00000000  1  VAR
43 00000000  1      COLOR      :ARRAY_TYPE0;
44 00000008  1      A,B        :ARRAY_TYPE1;
45 00000010  1      C          :ARRAY_TYPE2;
46 00000014  1      DAY        :DAY_OF_WEEK;
47 00000015  1      E,F        :ARRAY_TYPE3;
48 00000026  1      NEXT_COLOR:SCALAR_TYPE;
49 00000027  1      Q,R        :REC_TYPE;
50 0000005C  1      J,K        :BYTE;
51 0000005E  1      COUNT      :INTEGER;
52 00000062  1      X          :INT;
53 00000064  1      Y          :PTR;
54 00000068  1
55 00000068  1  $PAGE$
 ...
```

Figure 3-2. Listing of Example Pascal Program

```
 60 00000000  1  PROCEDURE INITHEAP(START,LENGTH:INTEGER);EXTERNAL;
  ...

 65 00000000  1  PROCEDURE PROC1 (VAR FPARM1:INTEGER; FPARM2:INTEGER);
  ...

 76 00000000  2      PROCEDURE RECURSIVE_PROC(VAR RP1:INTEGER; RP2:INTEGER;
 77 00000000  3                   VAR RP3:INTEGER; RP4:INTEGER; VAR RP5:INTEGER;
 78 00000000  3                   RP6:INTEGER; VAR PTR:REC_TYPE_PTR);
  ...

 91 00000004  3      BEGIN  (* RECURSIVE_PROC ENTRY *)
  ...

113 00000074  3          RECURSIVE_PROC (RP1,RP2,RP3,RP4,RP5,RP6,PTR);
  ...

125 0000015C  3      END;   (* RECURSIVE_PROC EXIT *)
  ...

128 00000164  2  BEGIN  (* PROC1 ENTRY *)
  ...

138 000001A6  2      RECURSIVE_PROC (FPARM1, FPARM1, FPARM2, FPARM2, D, D,
139 00000000  2                     REC_PTR^.NEXT_REC);
  ...

143 00000204  2  END;   (* PROC1 EXIT *)
  ...

146 00000000  1  PROCEDURE PROC2 (A:INTEGER);
  ...

153 00000000  2      PROCEDURE NESTED_PROC (PARM:INTEGER);
154 0000020C  3
155 0000020C  3      BEGIN (* NESTED_PROC *)
  ...

157 00000210  3      END;  (* NESTED_PROC *)
  ...

159 00000218  2  BEGIN (* PROC2 ENTRY *)
160 00000218  2      NESTED_PROC (A);
161 00000224  2  END;  (* PROC2 EXIT *)
  ...

183 0000023C  1  BEGIN  (* MAIN PROGRAM ENTRY *)
  ...

223 00000300  1      PROC1 (COUNT,COUNT+2);
224 00000312  1      PROC2 (COUNT+2);
  ...

235 00000390  1  END. (* MAIN PROGRAM EXIT *)
```

Figure 3-2. Listing of Example Pascal Program (Cont'd)

Defining A Default Path (Optional)

The next step in making a measurement with the software analyzer is defining the default path. The default path may be a module within a file or a file itself. The default path is used by the software analyzer when a command requires a path definition, but none is included in the command statement itself. For this measurement example, the default path is defined as the file MYFILE using the command:

setup default_path **MYFILE** RETURN

Therefore, for any commands being executed that do not include a file specification, the software analyzer will look for the defined parameters in the default path file MYFILE.

Specifying The Acquisition Depth (Optional)

The acquisition depth is user-definable with a default value of 256 states. An acquisition depth of up to 8192 states may be specified. For this measurement example, an acquisition depth of 100 states is specified using the following command:

setup depth **100** (RETURN)

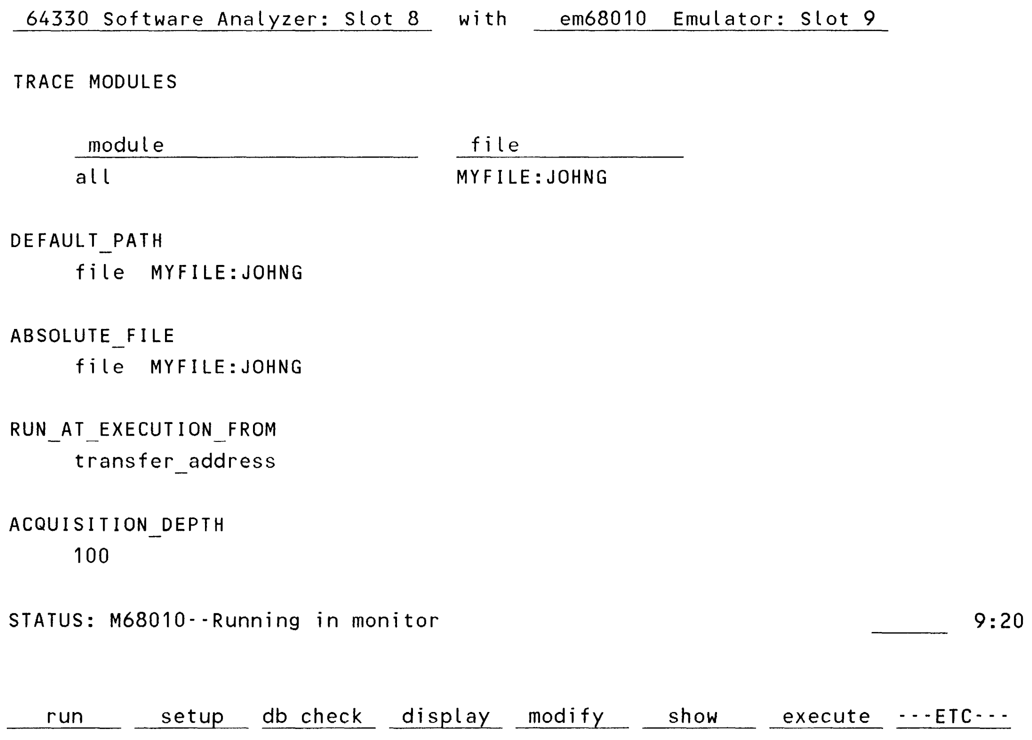
When a measurement is executed, the software analyzer will acquire and store 100 acquisition states in the trace file. For the trace modules measurement, there is a two-to-one correspondence between acquired states and displayed lines in the trace listing.

Setting Up The Trace Specification

The last step remaining before executing a trace measurement is setting up the trace specification. Entering the command

setup trace modules all (RETURN)

will cause the software analyzer to be configured to trace all modules in the default path file MYFILE. Had we wished to trace modules in a file other than the default path, we could have by adding the *file* parameter followed by the file name to the *setup* command. We have now set up the measurement and the complete setup display is shown in figure 3-3.



```
 64330 Software Analyzer: Slot 8   with   em68010  Emulator: Slot 9

TRACE MODULES

   module                             file
   all                               MYFILE:JOHNG

DEFAULT_PATH
   file  MYFILE:JOHNG

ABSOLUTE_FILE
   file  MYFILE:JOHNG

RUN_AT_EXECUTION_FROM
   transfer_address

ACQUISITION_DEPTH
   100

STATUS: M68010--Running in monitor                              9:20

   run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 3-3. Software Analyzer Setup Display

Interpreting The Trace Listing

Pressing the *execute* softkey followed by RETURN causes the software analyzer to initiate the trace modules measurement and start execution of the user program. After the specified 100 states have been acquired, the measurement stops, and the acquired data is processed and displayed on the screen. The trace modules listing is shown in figure 3-4.

The symbol field contains the names of the modules traced. In this example, all modules are Pascal procedures. The software analyzer looks up the module name in the compiler symbol file corresponding to the traced address value in the data record and displays that symbol in the first display field. In the status field, the analyzer display shows whether the traced address is the entry point to the module or the exit point from the module. To display the source field, the software analyzer looks up the source file line number contained in the assembler symbol file and extracts that line from the source file for display. The number "12" displayed in the status line indicates that the current trace line (displayed in inverse video in the center of the display) corresponds to acquisition state 12.

Looking at the program listing in figure 3-2, we see that the main program begins at line 183. The first module traced is PROC1, being called at line 223. The procedure INITHEAP defined at line 60 is a library function and is not included in the compiler symbol file for file MYFILE. Execution of PROC1 begins at line 128 whereas the line number displayed in the listing is 223, the line from which PROC1 was called. The called address of a module is considered the entry point.

The second line in the trace listing shows RECURSIVE_PROC being called at line 138. Note that RECURSIVE_PROC is indented one column, indicating that it is called from within PROC1. Nesting of modules are indicated by indentation. We see successive calls to RECURSIVE_PROC, each indented one column from the other, followed by successive exits from the module. This indicates that RECURSIVE_PROC is a recursive routine. This is verified by the program listing in figure 3-2. After execution of RECURSIVE_PROC, the program exits PROC1 and the main program then calls PROC2. PROC2, in turn, calls NESTED_PROC.

In this manner, the software analyzer provides an overview of program activity that enables you to quickly determine whether the program is executing modules in the sequence intended or, if not, in which module the program is in error. In the later case, the user can now use other software analyzer measurements to isolate the error more precisely.

SAVING THE CONFIGURATION

If you wish to retrieve the measurement setup for use at a later time, you need to save it in a configuration file. In this way you can begin to build a library of configurations and save a great deal of time in future measurement sessions. Pressing the *configure* and *save_in* softkeys in the sequence shown, typing in CONFIG1 and pressing the RETURN key will save the present configuration in a file named CONFIG1.

configuration *save_in* **CONFIG1** RETURN

This allows you to change your configuration (or end the session) with the assurance that you can retrieve your current configuration at a later time, if desired.

This completes the introduction to the software analyzer. You have seen how to load and execute a program with the emulation system and how to perform a simple measurement. For more specific and detailed measurements, refer to the information contained in the following chapters.

NOTE

In the measurement display examples given in this manual, the center line is always shown underscored. On your 64000 station, the center line is displayed in inverse video.

```
64330 Software Analyzer: Slot 8   with   em68010  Emulator: Slot 9

Symbol                  Stat   Source

PROC1                   entry  223 PROC1 (COUNT,COUNT+2);
 RECURSIVE_PROC         entry  138 RECURSIVE_PROC (FPARM1, FPARM1, FPARM2,
  RECURSIVE_PROC        entry  113 RECURSIVE_PROC (RP1,RP2,RP3,RP4,RP5,RP6,PTR);
   RECURSIVE_PROC       entry  113 RECURSIVE_PROC (RP1,RP2,RP3,RP4,RP5,RP6,PTR);
    RECURSIVE_PROC      entry  113 RECURSIVE_PROC (RP1,RP2,RP3,RP4,RP5,RP6,PTR);
    RECURSIVE_PROC      exit
   RECURSIVE_PROC       exit
  RECURSIVE_PROC        exit
 RECURSIVE_PROC         exit
PROC1                   exit
PROC2                   entry  224 PROC2 (COUNT+2);
 NESTED_PROC            entry  160 NESTED_PROC (A);
 NESTED_PROC            exit
PROC2                   exit

STATUS: Awaiting command                                      12      10:45


<STATE #>                            display   copy    show    end
```

Figure 3-4. Trace Modules Measurement Display

RECOMMENDED PROGRAMMING STYLE

The following programming style suggestions are recommended to achieve the best results from your analysis session.

1. Put only one statement on each line, especially when variables are used in more than one statement. The analyzer cannot determine which access has been made and only the first one may be displayed.

2. Break up compound statements, such as "IF <exp> THEN <stmt> ELSE <stmt>" to at least one line for each of the three parts.

3. Put comments on all "END" text to indicate to which structure it belongs. e.g. "END; /*FOR count LOOP*/"

4. Use BEGIN/END pairs on separate lines to mark all control structures and statements. This is redundant information in terms of compiler semantics and produces no additional code, but it clarifies the source display in the measurement analysis.

5. Put subroutine calls with all parameters on one line when possible.

Chapter 4

BUILDING DATABASE FILES

INTRODUCTION

The software analyzer has a high level of interaction with the HP 64000 compilers. This chapter describes the symbolic interface between the analyzer and compiler, and how the analyzer database is built when a program is compiled and linked. It also describes how to verify that the database file is correct. A list of compiler directives and the implications of their use with the software analyzer is also discussed.

SYMBOLIC INTERFACE

The software analyzer provides the capability for the user who has developed programs using the HP 64000 Logic Development System compilers, assemblers, and linker to specify measurements in terms of the symbols used in the programs. The compilers, assemblers and linker produce symbol tables that provide the analyzer with the information necessary to determine the physical addresses associated with the user's symbols.

The software analyzer accommodates both statically stored symbols (global variables and the names of software modules such as programs, functions, and procedures) and dynamically stored symbols (local variables, VAR parameters, and value parameters). The analyzer also allows you to reference source statement line numbers. The different symbol storage classifications and data types are explained in detail in chapter 11.

COMP_DB FILES

The basis of the software analyzer measurements are the comp_db (compiler database) files. The comp_db files allow the software analyzer to decode symbols into addresses and the addresses back into symbols. The comp_db files provide information on the symbol types (used for display purposes) and ownership of symbols by functions, procedures, or files.

Comp_db files can be generated in two ways; (1) by compiling the source file with option ***comp_sym*** and linking with option ***comp_db***, or (2) by using the ***generate_database*** utility. Due to the time required to build the comp_db files, it is suggested that you keep only a small working set of these files.

Since the linker creates a comp_db file for each comp_sym (compiler symbol) file it finds in the list of files being linked, old comp_sym files that are not being used should be purged. For each comp_sym file purged, the corresponding comp_db file should also be purged so that the software analyzer will not use a file that is not up-to-date. Conversely, when you make changes to files which are being tested and fail to either compile with the ***comp_sym*** option or link with the ***comp_db*** option, then unpredictable results can occur in the software analyzer measurements. The software analyzer has a ***database_check*** command that allows you to verify that all comp_db files are up-to-date. The ***database_check*** command is described later in this chapter.

BUILDING THE DATABASE FILE

The procedure for building the symbol database required by the software analyzer is described in the following paragraphs. The procedure is illustrated graphically in figure 4-1.

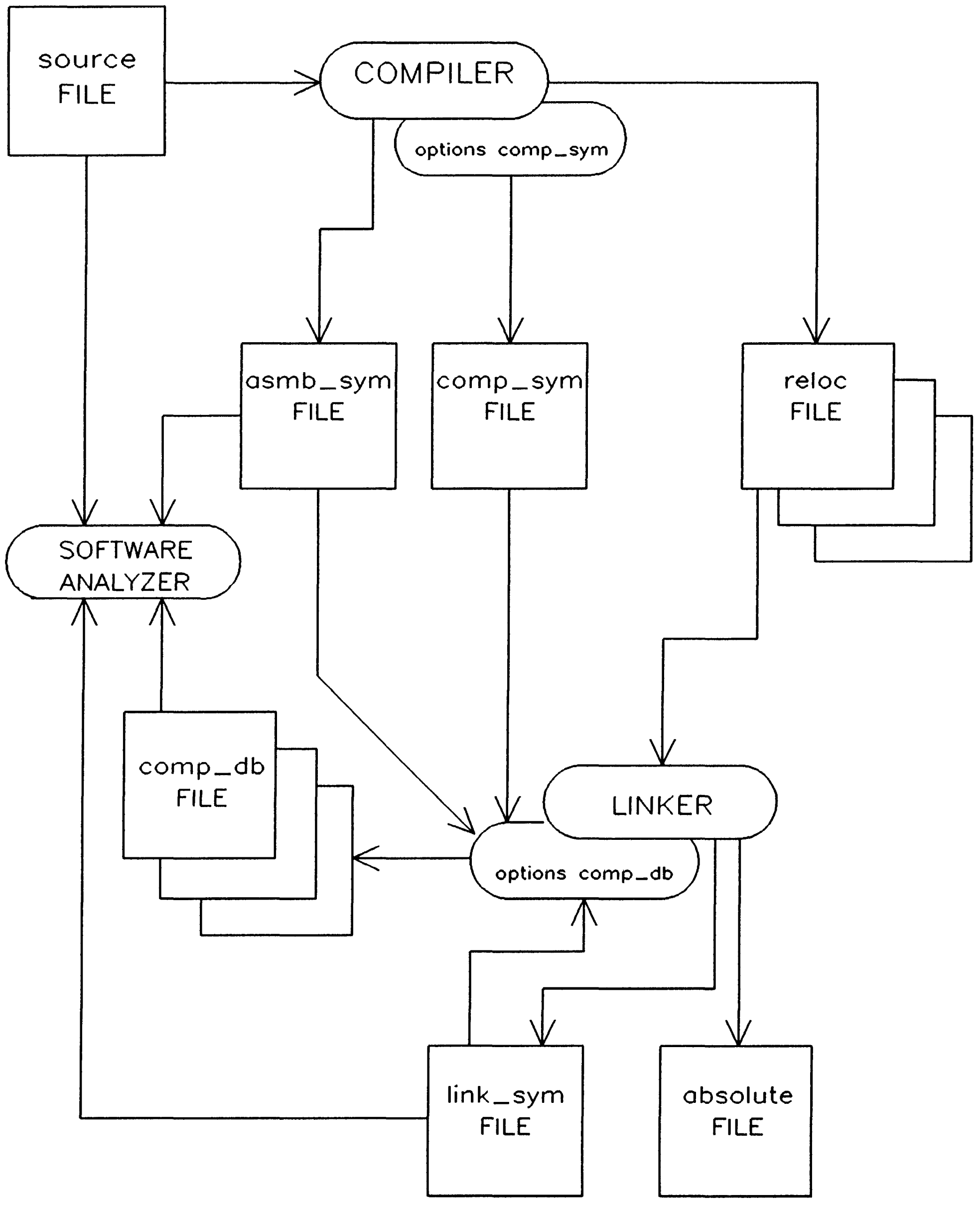


Figure 4-1. Software Analyzer Symbolic Interface

Compiling Files

All files that you wish to debug must be compiled with the ***comp_sym*** option which specifies the saving of the compiler symbol file. The command is:

compile <filename> options ... comp_sym

The "..." located between ***options*** and ***comp_sym*** signifies that other options may be specified in addition to the ***comp_sym*** option. However, the ***comp_sym*** option must be the last in the list of options.

COMPILER SYMBOL FILE. A compiler symbol file is generated for each file compiled on the HP 64000 system. The compiler symbol file includes the following information; the processor for which the file was compiled, the language the file was written in, the names, addresses, and data sizes for modules, and the names, types, sizes and locations of variables unique to each module. The compiler symbol file is not automatically saved after each compilation. The ***comp_sym*** option must be specified at compilation time which causes the compiler symbol file to be saved, making it accessible to the software analyzer.

ASSEMBLER SYMBOL FILE. An asmb_sym (assembler symbol) file is created for every file compiled unless the ***nocode*** option is included in the compile command. The contents of the asmb_sym file for a compiled file include local symbol names and relocatable or absolute addresses for those local symbols. Also, the addresses for line numbers are recorded here. In order for the analyzer to execute correctly, the asmb_sym file must be created for each file to be analyzed.

NOTE

DO NOT compile the file to be analyzed with the option ***nocode***. This will suppress the creation of an assembly symbol file, a file required for proper operation of the software analyzer.

Linking Files

Next, the compiled files must be linked. The command to use is:

link ... options ... comp_db

The link_sym file is created during the linking process and contains information about all files included in the ***link*** command. Included are global symbol names and their relocated addresses, source names and their relocated addresses, and a list of memory space used by the linked files.

NOTE

If local stack variables and static variables are to be traced in the same measurement, the stack should be assigned to a lower memory location than the static variables. Do **not** link the emulation monitor (Mon68010:HP) between the stack and static variables. This can result in erroneous data being displayed in the trace listings.

A data base file is created at link time, when ***options comp_db*** is specified, for each file that was compiled with the ***comp_sym*** option. The ***comp_db*** must be the last specified option in the link command, as ***comp_sym*** was in the compile command.

NOTE

When Pascal and C files are linked within the same absolute file, the linker will execute faster if the Pascal and C files are linked in separate blocks and not intermixed with each other.

Using The Generate_Database (gen_db) Command.

The generate_database Command allows you to generate comp_sym and comp_db type files without the overhead of recompilation and relinking. These files are optionally generated by the HP 64000 hosted compilers and linkers. The generate_database utility provides the capability to generate the comp_sym and comp_db type files for source files developed in HP supported hosted environments other than the HP 64000 environment. The generate_database utility provides the necessary link for performing high level software analysis in an HP 64000 development station of programs developed in the hosted development environment.

The utility can also be used to generate comp_db files for source files developed on an HP 64000 development station, but not compiled with the comp_sym option or linked with the comp_db option. This command first executes pass 1 of the compiler to generate the required comp_sym file. It then uses the asmb_sym, comp_sym, and link_sym files to generate the comp_db file. If a valid comp_sym file exists, you can specify that only the comp_db file be generated. The command then uses the existing comp_sym file, eliminating compiler pass 1 execution.

Specifying the keyword ***comp_sym*** allows you to perform syntax checking on a source program without the overhead of compiling the program.

NOTE

You do not need a compiler on your HP 64000 system to generate comp_sym and comp_db files using the ***generate_database*** command.

REQUIRED FILES. The generate_database utility requires the following files to be downloaded from the hosted development environment:

Pascal and C source files
Absolute files (.X)
Asmb_sym files (.A)
Link_sym files (.L)

A high level debug files transfer utility is available on the hosted system. This utility transfers all files required by the generate_database utility. See the Files Transfer Utilities section of the Hosted Development System User's Guide for detailed information on the transfer utility.

GENERATE_DATABASE COMMAND SYNTAX. The command syntax for the generate_database command is shown in figure 4-2.

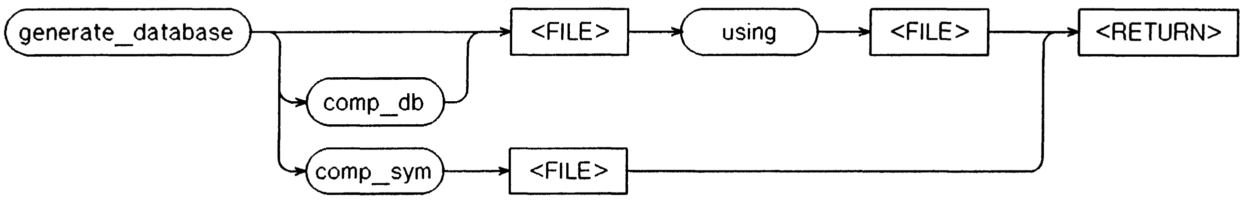


Figure 4-2. Generate_Database Command Syntax Diagram

GENERATE_DATABASE COMMAND PARAMETERS. The following parameters can be specified in the generate_database command.

comp_db	***comp_db*** specifies that only a comp_db file be generated.
comp_sym	***comp_sym*** specifies that only a comp_sym file be generated.
<FILE>	**<FILE>** must be the name of a C or Pascal source file for which the comp_sym and/or comp_db files are to be generated. In order for the database generator to distinguish between the two, C source files must have "C" for their first line and Pascal files must have "PASCAL" for their first line.
using <FILE>	***using*** <FILE> specifies the link_sym file (generated by a previous link) to be used in generating the comp_db file.

GENERATE_DATABASE COMMAND EXAMPLES. The following command examples illustrate how to use the generate_database command.

generate_database C_SOURCE ***using*** C_LINKSYM

The files C_SOURCE:comp_sym and C_SOURCE:comp_db are generated. C_SOURCE is a C source file and C_LINKSYM is a link_sym file generated by a previous link of C_SOURCE with other relocatables.

generate_database comp_sym PASCAL_SOURCE

The file PASCAL_SOURCE:comp_sym is generated.

generate_database comp_db PASCAL_SOURCE ***using*** PASCAL_LINKSYM

The file PASCAL_SOURCE:comp_db is generated.

VERIFYING DATABASE FILES

For proper operation of the software analyzer, the database information provided by the compiler and linker must be current for the files being analyzed. The ***database_check*** command is provided to systematically verify whether the database files associated with the current absolute file are up-to-date. The following paragraphs describe how database files are verified. The ***database_check*** command syntax is shown in figure 4-3.

The normal sequence of creating files is described in the previous section of this chapter, Building the Database Files. This process generates the following files in the order listed: 1) the comp_sym file, 2) the reloc file, and 3) the asmb_sym file. The reloc files are then linked with option ***comp_db*** specified. This generates, in order, the link_sym, absolute, and comp_db files.

During the normal operation of a link, the link_sym file will always be dated before or with the same date and time as the absolute file. The database_check function compares the modify date of the current absolute file with the modify date of the link_sym file associated with that absolute file. If the absolute file's modify date and time is earlier than that of the link_sym file, a database error exists. This error will be displayed and the database check will be terminated.

Since a relocatable file may be linked to any number of absolute files, the comp_db file for each file compiled with options comp_sym will contain the file name of the last absolute file it was linked to. With this information, the database_check performs the following operations:

1. Obtains the names of all linked files from the link_sym file.
2. Determines if the named files have comp_db files (i.e., if they were compiled with option ***comp_sym*** and linked with option ***comp_db***).
3. Verifies that comp_db files were generated for the current absolute file by comparing the name of the currently loaded absolute file with the absolute file name contained in the comp_db files.
4. Verifies that all files were generated in the proper sequence by comparing the modify date and time for each file type.
5. Reports any discrepancies in the database.

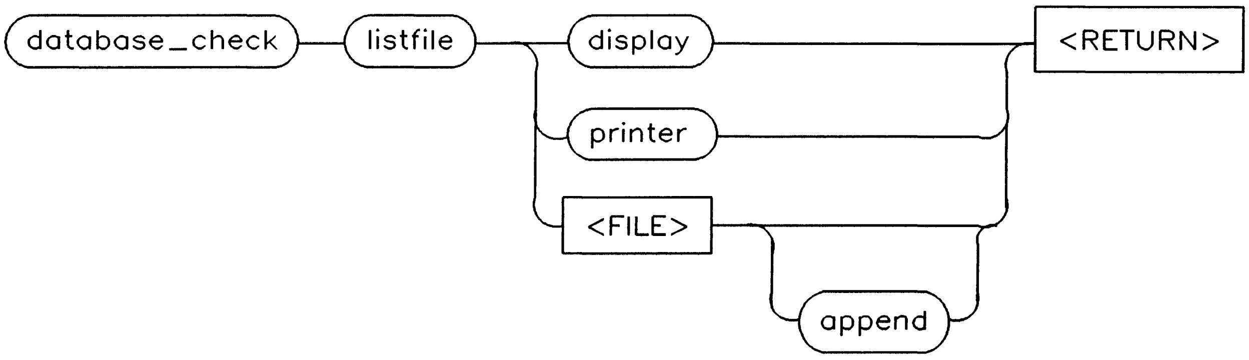


Figure 4-3. Database_check Command Syntax Diagram

The following example commands illustrate use of the ***database_check*** command:

```
database_check listfile display
database_check listfile printer
database_check listfile DATABASE_CHECK append
```

NOTE

To ensure that each file has the correct modify date, always keep your system clock set to the current date and time. This is done using the ***date&time*** utility command at the system monitor level of softkeys.

USING COMPILER DIRECTIVES

There are certain compiler directives that must be in the ON state for the Software Analyzer to operate correctly and others that may cause unexpected results.

AMNESIA

When the *AMNESIA* option is *OFF* there may be accesses to variables that could be missed because they are stored in registers.

ASMB_SYM

ASMB_SYM must be *ON*.

FIXED_PARAMETERS (C only)

When *FIXED_PARAMETERS* is *OFF* the software analyzer may display a parameter as being accessed when it was not. This occurs when the calling routine does not pass all the parameters to the called routine.

LINE_NUMBERS

LINE_NUMBERS must be *ON*.

OPTIMIZE

If the *OPTIMIZE* option is *ON* some accesses to variables may be missed because they are stored in registers.

FILES WRITTEN IN ASSEMBLY LANGUAGE

If a module is written in assembly language and you wish to trace a variable within the assembly language module, the variable must be declared as external in some other Pascal or C file. When the assembly language variable is not declared external, its address will not appear in the data base of any file and the analyzer will not be able to find the variable when a measurement is specified.

Chapter 5

CONTROLLING THE EMULATOR

INTRODUCTION

The software analyzer uses the emulation subsystem as an execution environment. The software analyzer includes a subset of the emulator commands to enable you to control emulation from within the analyzer. These commands are ***break, load, reset,*** and ***run***. These are the four basic commands needed to control a user's program running in emulation memory. The incorporation of the emulator commands simplify the interface between you and the system by providing the means for you to control the emulator without exiting the software analyzer.

EMULATION INTERFACE

Emulation Configuration File

When you invoke the software analyzer with the ***sw_anly*** command, you must specify a file name. This file name can be the name of a emulation command file or the name of a software analyzer configuration file. When both file types exist with the same file name, the software analyzer configuration file is used.

The emulation command file is the command file that was used in emulation to configure the emulator for a particular application. This file is generated during the emulation session and contains your answers to a series of questions ending with "**Command file name?**". This command file name is used to create a file of type "emul_com" (emulation command). The software analyzer uses this command file to determine which emulator and internal analyzer is used for software analysis. This configuration file is also used to determine the state of the emulator. When an emulation command file is specified, the analyzer is always reconfigured to the specified emulator. The current software analyzer configuration is lost.

The software analyzer configuration file is created with the ***configuration save_in*** command during a software analysis session. The software analyzer uses the configuration file to determine which emulator and internal analyzer is used for software analysis. The configuration file also configures the software analyzer for a measurement.

When no file is specified, the software analyzer prompts you with the question "**Emulation command file?**". you must specify a file before the system will allow you access to the software analyzer.

Loading And Running The Program

In order to use the software analyzer for debugging a program, the program must be loaded to the emulation system and running. This can be done within the software analyzer (refer to the descriptions of the emulation commands and their syntax in this chapter). After the emulation configuration is complete, load the absolute file into emulation memory using the ***load*** command.

If the address range into which a program is to be loaded resides entirely in internal emulation memory, the processor remains in the reset state. If any portion of the program resides in memory which has been mapped as external user memory, the processor is released from the reset state. After loading all portions of the file which are to reside in emulation memory, a handshake is performed to determine if the processor is executing in the emulation monitor program and, if not, a break is performed. When the processor is in the monitor, the user memory portion of the program is loaded. This sequence can be performed manually by using the options of the ***load*** command which specify the portion of memory to be loaded.

NOTE

When using the HP 64243,HP 64244, or HP 64245 emulators, the absolute file will be loaded to the last address space specified in the emulator, i.e. supervisor or user program space, supervisor or user data space, etc.

NOTE

When the emulator is running in the monitor, the processor must be reset before an absolute file containing the emulation monitor program can be loaded.

Once the program has been loaded, release the processor from the reset state with the ***run*** command. If the command is issued as the single keyword ***run***, the processor will use the start-up vector or routine to start execution in the emulation monitor. The status line will display "**Running in monitor Monitor initialized**" indicating that the HP 64000/monitor handshake is being performed. From this state you can issue the ***run*** command to begin execution of the program. When the command ***run*** is given, program execution begins at the transfer address specified in the source program. Thereafter, ***run*** will cause execution to begin at the address contained in the program counter (PC) register.

Selecting The Emulation Analysis Mode (HP 64243,64244,64245 Emulators only)

The emulator analysis mode must be set to ***bus_cycle_data*** in order to use the software analyzer. From within the emulation subsystem, execute the command ***modify analysis_mode_to bus_cycle_data***. If the emulation analysis mode is not set to ***bus_cycle_data***, the error message "**Incorrect analysis bus mode for this analyzer**" is displayed on the status line when you attempt to access the software analyzer.

COMMUNICATION BETWEEN THE SOFTWARE ANALYZER AND EMULATION

The software analyzer communicates with the emulation processor by transferring data to and from emulation memory. Data transfer is accomplished through the memory controller board into the emulation memory boards. The memory controller contains a hardware mapper that is programmed by the emulation command file to map the emulation processor address space into emulation or user memory spaces designated as RAM and/or ROM memory.

The software analyzer controls the emulation processor reset and break functions directly through the emulation control board. Refer to your HP 64000 System Emulation/Analysis manual for a more detailed description of how the HP 64000 host processor controls emulation.

The software analyzer and the emulator communicate the status of the emulator hardware to each other. Whenever the emulation hardware is modified by either the software analyzer or the emulator, the hardware change is reflected when the other module is entered. Note that the status of the hardware is communicated to the other module only if the modules are exited using the ***end*** softkey.

USING THE EMULATION MONITOR

The software analyzer makes extensive use of the emulation monitor. The emulation monitor must be linked with your program and must reside in emulation memory. The monitor supplied with the emulation software is designed to work with the software analyzer. If you have modified this monitor, it is possible that the software analyzer will not function properly. To verify that a modified monitor will function properly, perform the following procedure:

1. Access the emulator and load the absolute file containing the modified monitor.

2. Set the emulator such that it is running in the emulation monitor.

3. Verify that the ***display registers***, ***modify memory***, and ***display memory*** commands execute correctly.

4. Insert software breakpoints in your code using the ***modify software_breakpoints*** command.

5. Verify that program execution is transferred to the monitor and that the software break message is displayed at the correct address when a software breakpoint is executed.

6. Verify that the emulator's single step feature functions correctly.

All of the preceding features and functions must execute correctly to ensure proper operation of the software analyzer. If any of the above steps fail, modify your emulation monitor until the problem is corrected.

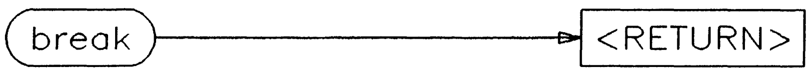
EMULATION COMMANDS

The following paragraphs will familiarize you with the emulation commands available in the software analyzer. A description of the command is given, including the syntax and examples of their use.

Break Command

The ***break*** command causes the processor to be diverted from execution of the user program to the emulation monitor. A break is defined as a transition from execution of a user's program to the Emulation Monitor.

SYNTAX



DEFAULT VALUE

none

EXAMPLE

break

PARAMETERS

none

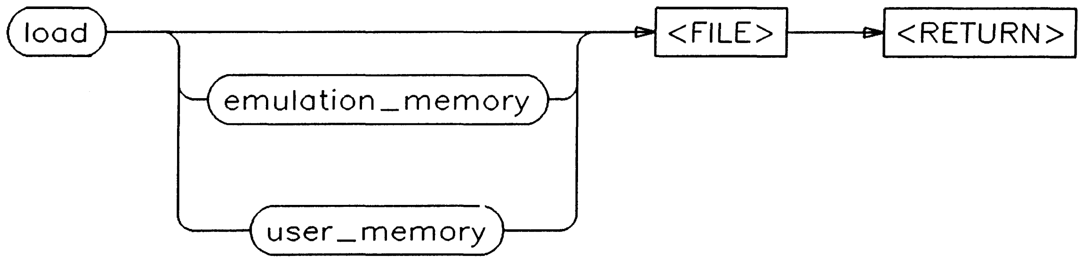
Load Command

The ***load*** command transfers absolute code from the HP 64000 system disc into user RAM or emulation memory. The destination of the absolute code is determined by the memory configuration map which was set up during emulation configuration and the address specified during linking. When using the HP 64243, HP 64244 or HP 64245 emulators, the absolute file will be loaded to the last address space specified in the emulator, i.e., supervisor or user program space, supervisor or user data space, etc.

NOTE

When the emulator is running in the monitor program and a ***load*** command is given which reloads the monitor, the results are unpredictable. If a reload of the monitor is required, first put the emulator in the reset mode.

SYNTAX



DEFAULT VALUE

all memory

EXAMPLES

load TESTP
load emulation_memory TESTP
load user_memory TESTP

PARAMETERS

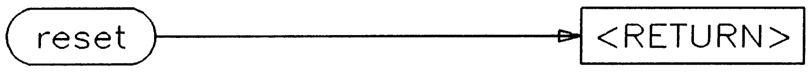
emulation_memory	***emulation_memory*** specifies that absolute code is to be loaded into emulation memory. The destination of the absolute code is determined by the address specified during linking.
<FILE>	<FILE> is the identifier of the absolute file to be loaded from the HP 64000 system memory into user RAM or emulation memory The syntax requirements for <FILE> are discussed in Appendix B.

user_memory	***user_memory*** specifies that the absolute program be loaded into user RAM in the target system. In the context of the load command, ***user_memory*** refers to target system memory.

Reset Command

The ***reset*** command suspends target system operation and reestablishes initial operating parameters, such as reloading control registers. The reset signal is latched when active and is released by the ***run*** command.

SYNTAX



DEFAULT VALUE

none

EXAMPLE

reset

PARAMETERS

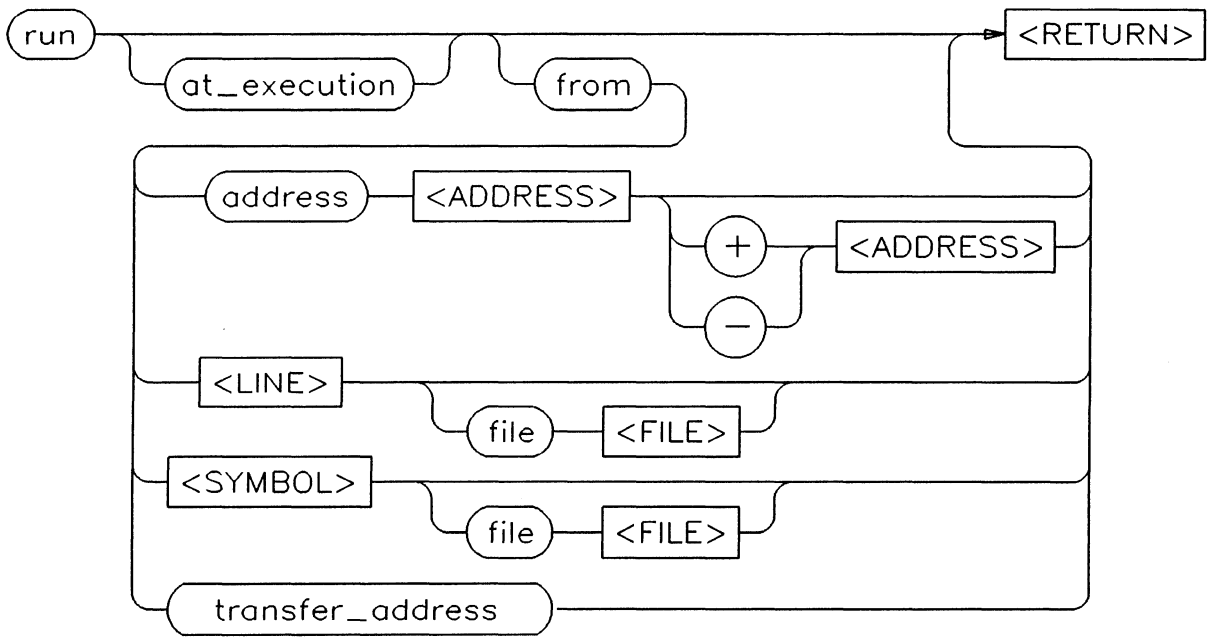
none

Run Command

When the processor is in a reset state, ***run*** causes the reset to be released, and if a ***from*** address is specified, the processor is directed to that address. If the processor is running in the emulation monitor, the ***run*** command causes the processor to exit into the user program. The program can either be run from (1) the transfer address of the user's program, (2) a specified address, (3) a specified line number in the source code, (4) from the entry point of a specified module, (5) from the address currently stored in the processor's program counter, or (6) from a global symbol.

A "***run at_execution***" command causes the user's program to start running after the ***execute*** softkey has been pressed. This enables the trace measurement to be started before beginning program execution, ensuring that the analyzer can trace all code executed starting with the "***run from***" location.

SYNTAX



DEFAULT VALUE

If no ***from*** option is specified with the ***run*** command, the emulator will begin program execution at the current address specified by the processor's program counter.

EXAMPLES

run
run at_execution from transfer_address
run from 173 file TESTP
run from PROC2
run at_execution from address 356AH
run from address 3490H + 0FFH

PARAMETERS

<ADDRESS>	<ADDRESS> represents an address within the absolute file loaded into user or emulation memory from which the processor will begin program execution. The syntax allows specification of a positive or negative offset from the absolute address.
<FILE>	<FILE> represents the name of the source file containing the address, line, or symbol from which the processor is to begin program execution.
<LINE>	<LINE> allows you to specify a line number in the source code as the starting point for program execution. Program execution begins at the absolute address containing the first executable instruction associated with the source line.
<SYMBOL>	<SYMBOL> allows you to specify program execution to run from a specified symbol. If a file name is specified with <SYMBOL>, the analyzer assumes that the symbol is a module in the specified file. If no file is specified with <SYMBOL>, the analyzer first looks for the address of a global symbol in the link_sym file associated with the currently loaded absolute file. If no global symbol is found there, the analyzer then searches for a module in the current default file.
<OFFSET>	<OFFSET> allows you to specify an positive or negative offset from the specified <ADDRESS>.

NOTES

Chapter 6

MAKING TRACE MEASUREMENTS

INTRODUCTION

This chapter describes the software analyzer trace measurement modes. These modes are (1) trace data_flow , (2) trace modules, (3) trace statements, and (4) trace variables. The uses of these modes are discussed in chapter 1 and syntax diagrams for each operating mode are given in this chapter and in appendix A, figures A-4 through A-7. This chapter provides general information concerning the operating modes, discusses the softkeys used to initiate each operating mode, and details the options for each mode.

If you have any difficulties or problems when using the software analyzer, see appendix E, Resolving Measurement Problems, for possible solutions.

GENERAL

Before setting up any of the measurement configurations discussed in this chapter, the program to be analyzed must be compiled using the ***comp_sym*** option and linked with the emulation monitor program and appropriate library routines using the ***comp_db*** option. The emulator must also be properly configured with the linked absolute file loaded in emulation memory. These procedures are described in chapter 3, Getting Started.

Before setting up a measurement, you may define a default path and set up the acquisition depth. The default path may be a source file name or a source file name along with a procedure or function name within the source file. The default path is used when a path is needed for a measurement and none has been included in the measurement command itself. Using the default path to specify a particular area of software activity can simplify the setup commands used during a measurement session since the path parameters in some commands can then be omitted.

The acquisition memory depth can be set to a value from 1 to 8192 states. The word "state" as used here and throughout this manual refers to logical states with respect to the analyzer and not physical states in the time domain. In general one byte of memory data corresponds to one logical state. The analyzer acquires these states by reading the target processor's memory or by capturing bus cycles. Each bus cycle that is acquired corresponds to two states. This will be true even if the bus cycle contained only one byte of valid data. Invalid data bytes are counted but otherwise ignored by the analyzer.

THE TRACE FILE

Each measurement that produces a display outputs raw trace data to a file known as a trace file. The trace file allows user-selectable trace depth for the measurements and provides the ability to review old measurements. A unique name for the trace file is created for each HP-IB address and analysis slot number used to generate a trace. The trace file name is

Swtrace<analysis slot No.><HP-IB address>:<userid>:trace

where <analysis slot No.> is the card cage slot occupied by the HP 64302A Analyzer

<HP-IB address> is the address of the development station on the system bus

and <userid> is the currently selected userid

Example: Swtrace76:USER1:trace

The file has two basic parts. The first part is the symbol header which contains information identifying what type of measurement was done as well as database information about the symbols used in the measurement. When a variable's database information is written to the symbol header, information describing all data types defined within a variables declaration is written to the file. There is a maximum size limit for this symbol data. If the requested variable or variables exceed this size, an error is displayed. Only symbols made up of very large numbers of different data types can cause this to occur. Reducing the number of symbols traced will help reduce this symbol data to below the maximum limit. Tracing a subset of the variable will have no effect since all symbol data is required to extract any subelement.

The second part of the file is the measurement data itself. When a ***display*** or ***execute*** command is given to the analyzer, the last measurement's Swtrace file is deleted and a new Swtrace file created. If you want to save a measurement, you must do it before the next ***display*** or ***execute*** command is given.

MAKING SEQUENTIAL TRACE MEASUREMENTS

At the completion of a measurement, the software analyzer causes the user program to break and the emulator to enter the emulation monitor. When a ***run*** or ***run at_execution*** command is given, the user program processor data is restored and program execution is resumed from the point that the break occurred. This feature enables sequential measurements to be made. Executing the commands ***run at_execution*** and ***execute*** causes the software analyzer to begin acquiring trace data in the current measurement at the point of execution where the previous measurement ended. This allows you to sequentially view segments of program activity in increments defined by the value set up in the acquisition_depth specification. The ***run_at_execution*** command remains enabled until another run command is executed.

MEASUREMENT CONSIDERATIONS

When a trace variables measurement with local variables is executed and the program is running in the monitor, the following condition applies:

> If ***run at_execution*** from current program counter is specified with the program currently running in the monitor and the next instruction to be executed is within the procedure where activity to be traced will occur, tracing will begin immediately upon exit from the monitor.

If the preceding condition is not met, i.e., the next program counter is outside the procedure to be traced or a ***run at_execution*** from an address has been specified, no data will be captured until the entry point of the procedure being traced has been detected. In this case, the software analyzer will display the message "**exit captured without matching entry information**" in the trace list.

The trace statements and trace data_flow measurements will always continue from where program execution left off, regardless of the measurement setup.

NOTE

The software analyzer does not distinguish between supervisor and user modes. If these modes are used to map multiple physical addresses to one logical address, the software analyzer will correlate all physical addresses with the last program loaded. This will probably result in erroneous data being displayed in the measurement display.

MEASUREMENT SYSTEM INTERFACE

The software analyzer cannot participate in an intermodule bus measurement. If an intermodule bus measurement is in progress with other analyzers, the IMB measurement must be halted before the software analyzer can be used.

The Internal Analyzer (HP 64302A) can be used for assembly level debugging in conjunction with the software analyzer for high level software debugging. This allows you to take an assembly level trace in emulation, break the processor at a specific point, invoke the software analyzer and make a high level measurement from where the assembly level left off.

SIMULATED I/O

The software analyzer does not support simulated I/O. When a simulated I/O request is made from the target processor while a measurement is in process, the measurement will be terminated and the message "**Simulated I/O Requested**" will be displayed. The target processor will be running in the simulated I/O routine waiting for a response from the host processor. To complete the I/O operation, ***end*** out of the software analyzer and enter the emulator. This will cause the I/O operation to complete and the program will continue running.

TRACE DATA_FLOW MEASUREMENT

The trace data_flow measurement traces the values of specified variables or parameters on entry to and exit from selected procedures or functions in a program. The traced variables must be accessible at the procedure entry point, exit point, or both. Static variables can always be accessed. Local variables can be accessed only if the variable (1) belongs to a parent procedure, (2) is a pass-by-reference parameter to the specified procedure, or (3) is a pass-by-value parameter and measurement is with procedure entry.

If the variable is local to the associated module, it can never be accessed since none of a module's local variables are created until after module entry and they are removed from the stack before module exit. Value parameters are active only at procedure entries, and reference parameters are always active with respect to their procedure. If a value parameter is requested on exit to its procedure, a warning message will be displayed. and no values of that parameter will be displayed.

When tracing recursive modules, the analyzer will stop capturing data if the number of recursive levels exceeds 128. A message is displayed in the trace list indicating that the recursion limit has been exceeded. Once the program returns to the 128th level of recursion, the capture of data is resumed. The count of recursive levels is relative to the start of the trace. If the trace was started in the middle of recursion (i.e., some recursive entries have occurred before the trace was started), it is possible for the analyzer to miss data. The analyzer will trace only recursive levels for which it finds entries. If the analyzer finds more exits than entries, a message is displayed in the trace list indicating that exits without matching entries have been detected.

Command Syntax

Up to ten symbols may be specified in the *setup trace data_flow* command in combinations of procedures (functions) and variables. For example, the setup command could call for nine variables to be traced in one procedure, or for four variables to be traced in each of two procedures. The command syntax for setting up the trace data_flow measurement is shown in figure 6-1.

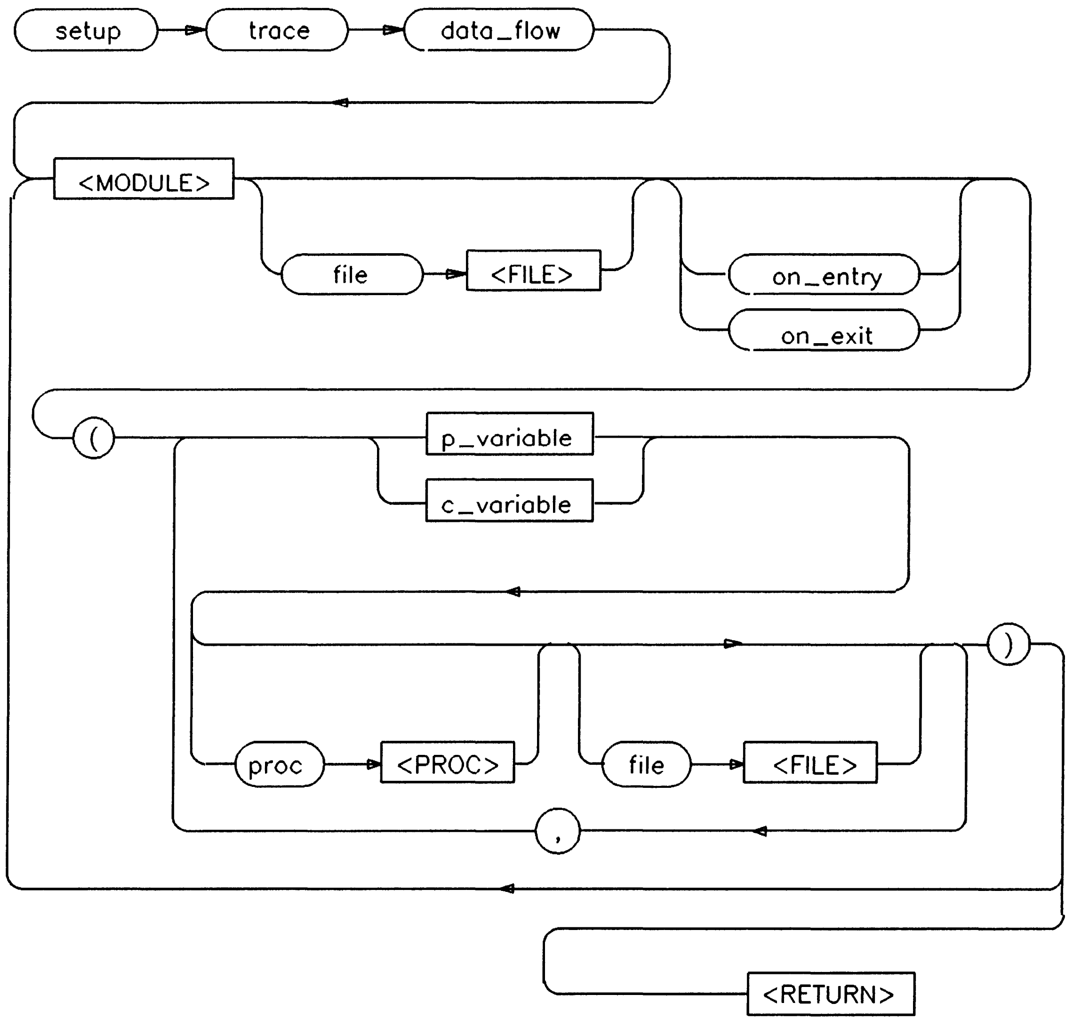
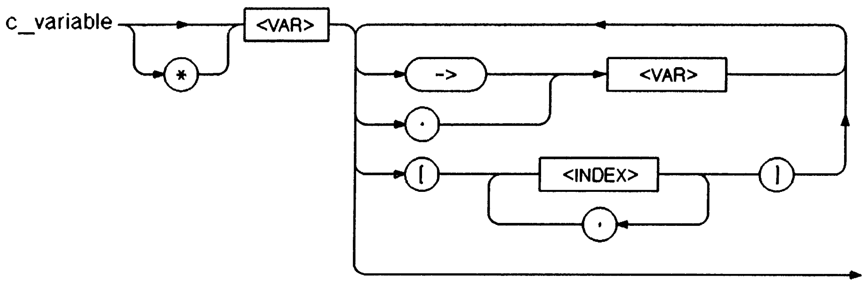


Figure 6-1. Setup Trace Data_Flow Syntax Diagram

Parameters

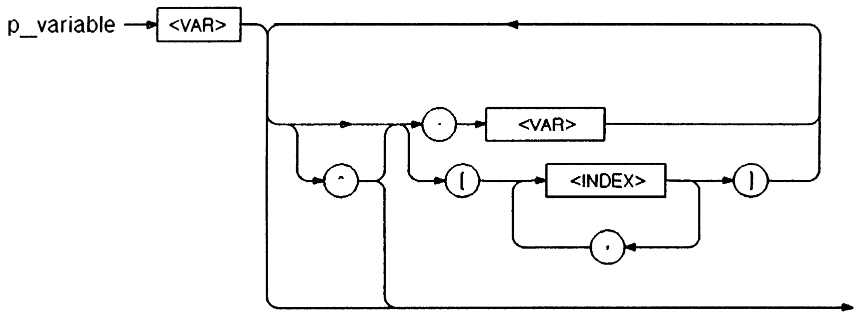
The following paragraphs define the parameters used in the ***setup trace data_flow*** command.

c_variable	c_variable may be any valid C variable.



on_entry	***on_entry*** specifies that data be traced only on entry to the specified module(s). The default value is to trace data on both entry to and exit from a module.
on_exit	***on_exit*** specifies that data be traced only on exit from the specified module(s).
<FILE>	<FILE> is an optional parameter that refers to the source file containing the specified <MODULE>, <VAR>, or <PROC> called out in the command statement. If the <MODULE>, <VAR>, or <PROC> is in the defined default path, the <FILE> parameter may be omitted from the command statement.
<INDEX>	Represents an index value (integer or scalar value) specifying a component of an array.
<MODULE>	<MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure or a function within a specified file. In C, a module can be the name of a function within a specified file. The trace data_flow measurement traces the specified variables on entry to and/or exit from the <MODULE> as specified in the command line. A given module can only be specified once.
<PROC>	<PROC> is an optional parameter that refers to a procedure or function in which <VAR> is declared. If <PROC> is defined in the ***setup default_path*** command, it may be omitted in the ***setup trace data_flow*** command. If <PROC> is not specified in either the default path or the ***setup trace data_flow*** command, the analyzer assumes that <VAR> is a global variable defined at the main program level.

p_variable — p_variable may be any valid Pascal variable.



<VAR> — <VAR> represents the name of a variable or parameter to be traced on entry to and/or exit from a <MODULE>. <VAR> can be any valid Pascal or C variable expression.

Examples

The following command examples illustrate how to use the ***setup trace data_flow*** command to define measurements.

> ***setup trace data_flow*** PROC2 (COUNT , D ***proc*** PROC1 ,
> PTR ***proc*** PROC2)
> ***setup trace data_flow*** PROC1 (SN ***proc*** PROC1 , SV ***proc*** PROC1)
> PROC2 (SNN ***proc*** PROC2 , SVN ***proc*** PROC2 , COUNT)
> ***setup trace data_flow*** PROC1 (AR[1,2,3] , RC.E1.EZ)
> ***setup trace data_flow*** PROC1 (A^.B^.C^)
> ***setup trace data_flow*** proc2 (*a->b->c)

The example below shows several lines of a program, a ***setup trace data_flow*** command and a portion of the resulting trace list.

SOURCE PROGRAM LINES.

```
101  PROCEDURE PROC4(A:INTEGER);
...
122  PROCEDURE PROC10(XV:INT; VAR XN:INT, YV:PTR, VAR YN;PTR)
```

SETUP MEASUREMENT COMMAND.

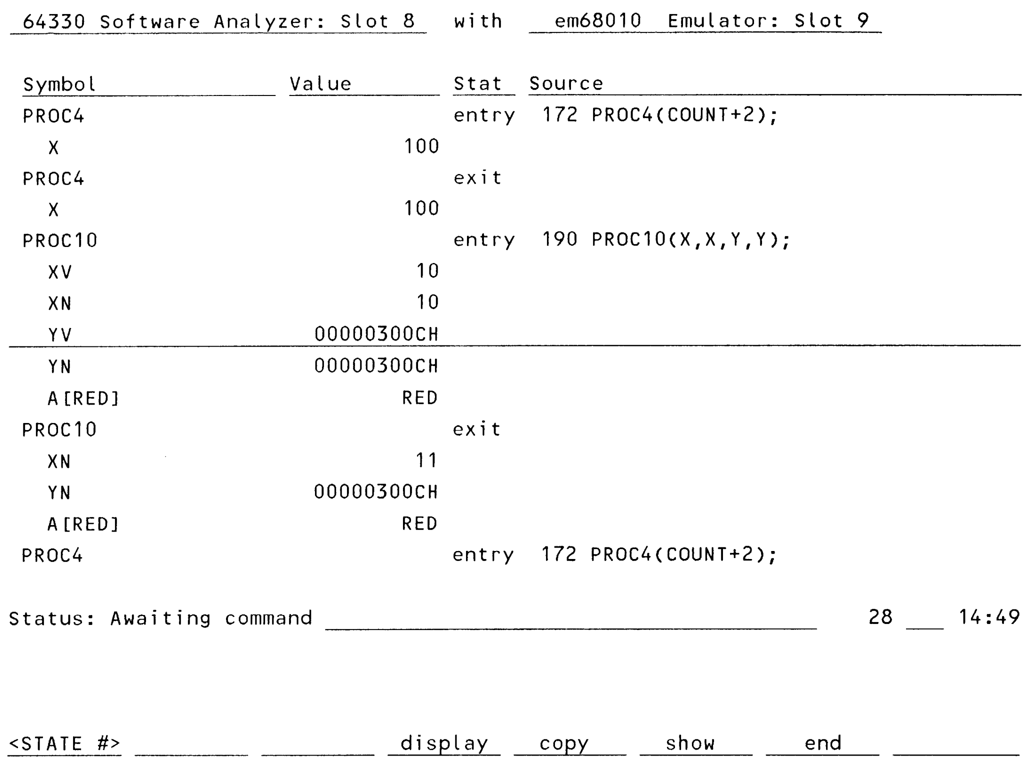
> ***setup trace data_flow*** PROC10 (XV ***proc*** PROC10, XN ***proc*** PROC10,
> YV ***proc*** PROC10, YN ***proc*** PROC10, A[RED]) PROC4 (X)

MEASUREMENT DISPLAY. The measurement display resulting from the preceding setup specification is shown in figure 6-2. The trace list shows that PROC4 is called from line 172 and PROC10 is called from line 190. The values of the variables are shown immediately following the entry or exit of the corresponding module. Note that when PROC10 is exited, XV and YV are not active and are not displayed.

Note also that YV and YN are pointers and their values are the values of the pointers themselves, not the objects of the pointers. The software analyzer will trace the object of a pointer. If the variable YV^ (*yv in C) had been specified, the value of the object pointed to by pointer YV would have been displayed in the trace list.

In the C programming language, array parameters without an explicitly defined size will not be traced as a whole.

Source lines displayed by trace data_flow measurements are not affected by instruction prefetch mechanisms. Source lines are not shown for exits.



```
 64330 Software Analyzer: Slot 8   with   em68010  Emulator: Slot 9

 Symbol               Value        Stat   Source
 PROC4                             entry   172 PROC4(COUNT+2);
   X                          100
 PROC4                             exit
   X                          100
 PROC10                            entry   190 PROC10(X,X,Y,Y);
   XV                          10
   XN                          10
   YV                   00000300CH
   YN                   00000300CH
   A[RED]                     RED
 PROC10                            exit
   XN                          11
   YN                   00000300CH
   A[RED]                     RED
 PROC4                             entry   172 PROC4(COUNT+2);

Status: Awaiting command                                  28     14:49


<STATE #>                      display   copy    show    end
```

Figure 6-2. Trace Data_Flow Measurement Display

TRACE MODULES MEASUREMENT

The trace modules measurement provides an overview of a program's control flow at the module level. This measurement provides the capability to isolate a problem to a specific module and to provide a history of the module calls leading up to the problem. The entry and exit points of procedures and functions are traced and displayed with indentation used to indicate the level of nesting of the traced modules. The measurement can be set up to measure all modules or selected modules within a file or group of files. A total of 128 modules can be traced.

When tracing recursive modules, each successive level of recursion is indented in the trace list. For large numbers of recursion levels, this may result in the data being shifted off the right side of the trace list display. This is indicated by an asterisk (*) displayed in the last display column. Recursive modules are indented relative to the outermost recursion level traced.

If a module name is longer than the symbol field width or the recursion level is deep, an asterisk is displayed in the last column of the symbol field. To display the entire name of a module, increase the symbol field width using the ***display*** command.

Command Syntax

The command syntax for setting up the trace modules measurement is shown in figure 6-3.

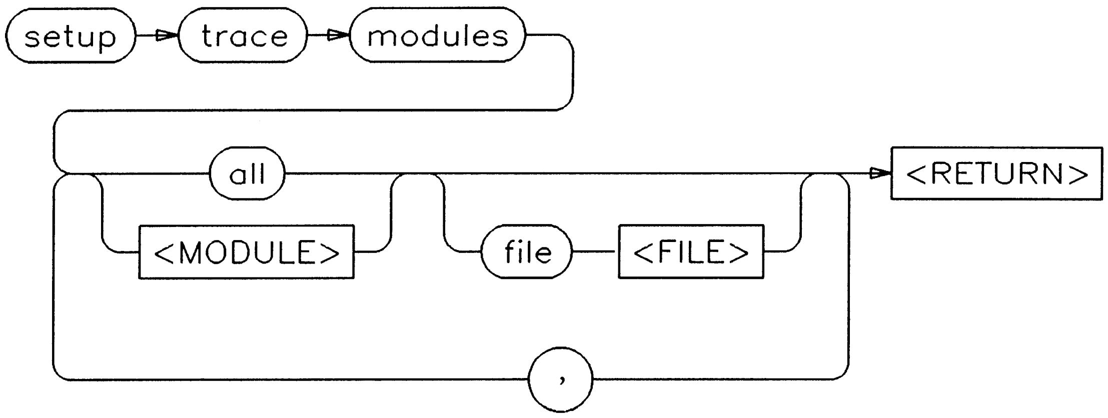


Figure 6-3. Setup Trace Modules Syntax Diagram

Parameters

The following definitions describe the parameters used in the ***setup trace modules*** command.

all	***all*** specifies that all modules in the designated file or default path be traced. A maximum of 128 modules may be traced.
<FILE>	<FILE> is an optional parameter that refers to the source file containing the specified modules called out in the command statement. If the

<MODULE> is in the defined default path, the <FILE> parameter may be omitted from the command statement.

<MODULE> <MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure or a function within a specified file. In C, a module can be the name of a function within a specified file. The trace modules measurement traces the entry and exit points of the specified modules.

Examples

The following command examples illustrate how to use the ***setup trace modules*** command to define measurements.

setup trace modules all
setup trace modules all file BSORT ***, all file*** TESTP ,
PROC1 , PROC2
setup trace modules PROC1 , PROC4 , PROC5

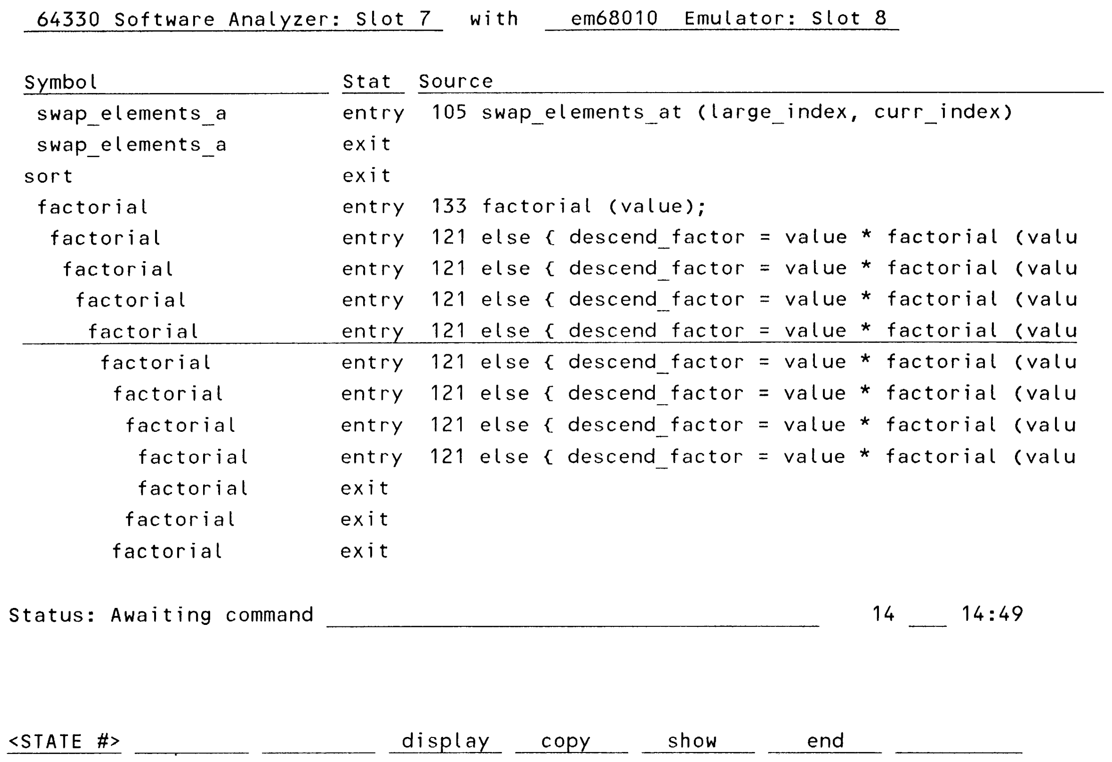
The following example shows a setup trace modules command and a portion of the resulting trace list.

SETUP MEASUREMENT COMMAND.

setup trace modules all file Util, *all file* Fact

MEASUREMENT DISPLAY. The trace list in figure 6-4 shows the sequence of entries and exits for all modules in source files Util and Fact. In this example procedure factorial is recursive. The recursive descent can be seen in the succession of "entry"s in the Status field and also by the indentation of the procedure name in the Symbol field. The recursive ascent is shown in a similar manner.

Source lines displayed by trace modules measurements are not affected by instruction prefetch mechanisms. Source lines are not shown for exits.



```
 64330 Software Analyzer: Slot 7   with   em68010  Emulator: Slot 8

Symbol                   Stat  Source
 swap_elements_a         entry  105 swap_elements_at (large_index, curr_index)
 swap_elements_a         exit
sort                     exit
 factorial               entry  133 factorial (value);
  factorial              entry  121 else { descend_factor = value * factorial (valu
   factorial             entry  121 else { descend_factor = value * factorial (valu
    factorial            entry  121 else { descend_factor = value * factorial (valu
     factorial           entry  121 else { descend_factor = value * factorial (valu
      factorial          entry  121 else { descend_factor = value * factorial (valu
       factorial         entry  121 else { descend_factor = value * factorial (valu
        factorial        entry  121 else { descend_factor = value * factorial (valu
         factorial       entry  121 else { descend_factor = value * factorial (valu
         factorial       exit
        factorial        exit
       factorial         exit

Status: Awaiting command                                         14      14:49


<STATE #>                        display   copy    show    end
```

Figure 6-4. Trace Modules Measurement Display

TRACE STATEMENTS MEASUREMENT

The trace statements measurement gives you a detailed view of a small section of code. Once a problem is isolated to a particular module, the trace statements measurement allows a close inspection of this code to determine precisely the problems. The measurement traces the execution of source language statements and the value of all variables in each statement in a defined source program line range or a specified module. Each source statement is displayed with its line number and the value of any variables referenced in the source statement.

When tracing recursive modules, the analyzer will stop capturing data if the number of recursive levels exceeds 128. A message is displayed in the trace list indicating that the recursion limit has been exceeded. Once the program returns to the 128th level of recursion, the capture of data is resumed. The count of recursive levels is relative to the start of the trace. If the trace was started in the middle of recursion (i.e., some recursive entries have occurred before the trace was started), it is possible for the analyzer to miss data. The analyzer will trace only recursive levels for which it finds entries. If the analyzer finds more exits than entries, a message is displayed in the trace list indicating that exits without matching entries have been detected.

Some processors prefetch instructions prior to their execution. Prefetches have the following effects on the trace statements measurements:

1. Accesses to variables by instructions executed immediately prior to the address range being traced will appear as accesses occurring within the address range.

2. Accesses to variables by the last instructions executed within the address range being traced may not appear.

3. The symbol and value fields in the display may be offset from their corresponding source lines.

The trace depth of a trace statements measurement may be greater than the depth specified in the setup menu. This due to the fact that the analyzer captures data in blocks of 512 bytes when executing a trace statements measurement.

The error message "break while in range, possible missing data" will occasionally appear on screen. This happens whenever the analyzer executes a break and the last opcode detected is in the range being traced. Data may be missing if the analyzer did not break on completion of an opcode. Data acquisition will continue at the start of the next opcode.

The trace statements measurement only displays variables accessed by the statements being traced. Any accesses caused by procedures or functions outside of the traced range are not shown. For example, if a variable is modified by a compiler library, that variable will not appear in the trace statements trace list.

Command Syntax

The command syntax for setting up the trace statements measurement is shown in figure 6-5.

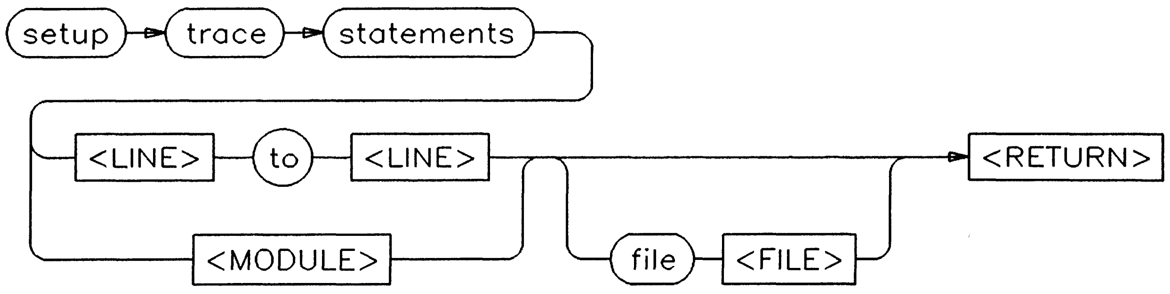


Figure 6-5. Setup Trace Statements Syntax Diagram

Parameters

The following definitions describe the parameters used in the ***setup trace statements*** command.

<FILE>	<FILE> is an optional parameter that refers to the source file containing the specified <MODULE> or line range called out in the command statement. If the <MODULE> or line range is in the defined default path, the <FILE> parameter may be omitted from the command statement.
<LINE>	<LINE> represents the line number of a Pascal or C statement in the source program. The two line numbers specified are the boundaries of the trace measurement. The first line is inclusive and is traced. The second line (following the ***to*** in the measurement specification) is noninclusive and is not traced. If the specified <LINE> contains only comments (no executable code), the analyzer will associate the line number with the first line containing executable code following it. Any comment lines preceding the first line of executable code in a procedure or function are not recognized by the software analyzer. All lines in the specified line range must be contained within a single module. This module may be a procedure or function, or the main program block.
<MODULE>	<MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure or a function within a specified file. In C, a module can be the name of a function within a specified file. The trace statements measurement traces source level statements and all variables referenced in the source statements contains in the specified <MODULE>.

Examples

The following command examples illustrate how to use the ***setup trace statements*** command to define measurements.

setup trace statements PROC2 ***file*** TESTP
setup trace statements 74 ***to*** 102

The following example shows several lines of a Pascal program, a ***setup trace statements*** command and a portion of the resulting trace list.

SOURCE PROGRAM LINES.

```
88      P2:=DN;   (*       DYNAMIC     CALLBYNAME*)
89      Y:=2;
90      A[ORANGE]:=A[RED];
91      C[-1]:=J+K;
92      PROC13;
93      (*Comment lines are not traced*)
94      X:=PROC1(COUNT,MAX);
95      Q.REAL_NUMBER:=2.0E22;
96      Q.SETT         :=[RED,WHITE,BLUE];
97      J              :=Q.N;
98      Q.VARIANT1:=A;
```

SETUP MEASUREMENT COMMAND.

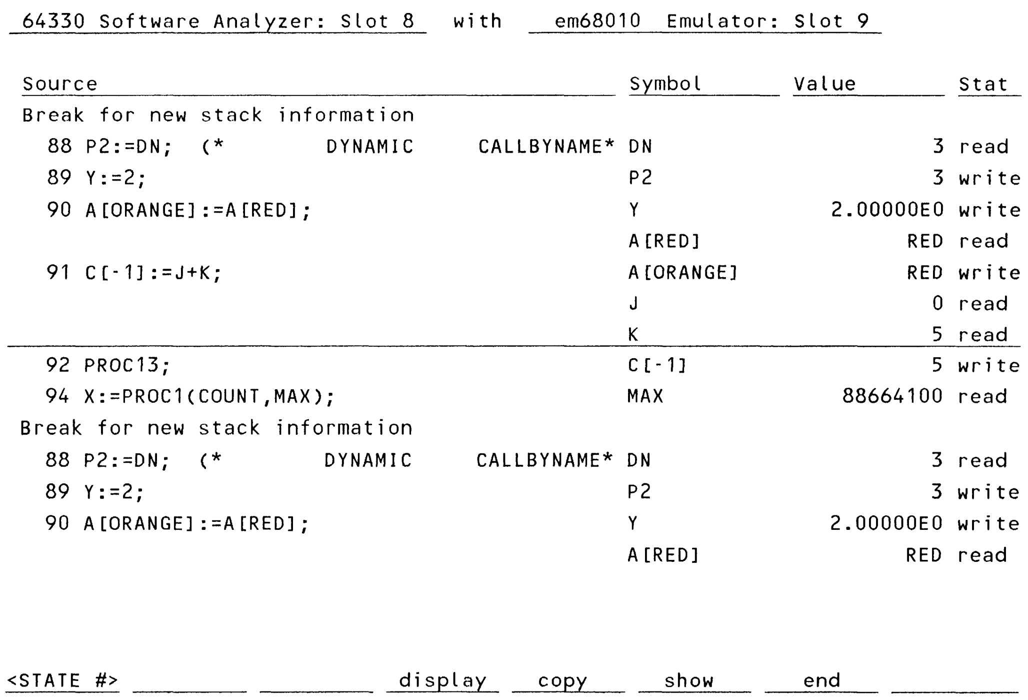
setup trace statements 88 ***to*** 95 ***proc*** SETUP ***file*** MAIN

MEASUREMENT DISPLAY. Figure 6-6 is a trace statements measurement listing showing the source lines that were executed and and the values of the variables accessed or modified in the source lines. "Break for new stack information" indicates that the analyzer has started tracing a different occurrence (activation) of the procedure SETUP.

Some variables accessed in a source line are not traced. This includes variables that are maintained in registers rather than in memory. This may occur if the compiler ***AMNESIA*** option is off. In the C programming language, array parameters without an explicitly defined size are not traced. The value of a pointer variable is traced but the object of the pointer is not traced.

Symbols may not line up with the source line that accessed them. This is seen in the sample display in figure 6-6. Executed source lines may not be displayed if the number of program fetches for the source line is less than the depth of the prefetch queue. The preceding limitations to the measurement apply only when the analyzer is used with a target processor which has an instruction prefetch mechanism and an emulator that does not dequeue the prefetch.

In a prefetch environment, the source line may be off by plus or minus one line or variables may be displayed with the wrong source line.



```
64330 Software Analyzer: Slot 8   with   em68010  Emulator: Slot 9

Source                                              Symbol      Value       Stat
Break for new stack information
  88 P2:=DN;  (*       DYNAMIC     CALLBYNAME*  DN                   3 read
  89 Y:=2;                                      P2                   3 write
  90 A[ORANGE]:=A[RED];                         Y           2.00000E0 write
                                                A[RED]             RED read
  91 C[-1]:=J+K;                                A[ORANGE]          RED write
                                                J                    0 read
                                                K                    5 read
  92 PROC13;                                    C[-1]                5 write
  94 X:=PROC1(COUNT,MAX);                       MAX           88664100 read
Break for new stack information
  88 P2:=DN;  (*       DYNAMIC     CALLBYNAME*  DN                   3 read
  89 Y:=2;                                      P2                   3 write
  90 A[ORANGE]:=A[RED];                         Y           2.00000E0 write
                                                A[RED]             RED read

<STATE #>                       display   copy   show   end
```

Figure 6-6. Trace Statements Measurement Display

TRACE VARIABLES MEASUREMENT

The trace variables measurement traces specified variables and parameters and displays their values, along with the source statement that accessed them. The variables are displayed in their declared data type format, i.e., as integers, reals, boolean values, characters, etc. The variable must be uniquely defined as to the module where it is declared. If the variable is defined outside a module, i.e., a program variable in Pascal or an outer level variable in C, then only the file name is required. When multiple variables map to the same memory location, only the first variable specified in the setup command is displayed. A maximum of 10 variables may be traced.

When tracing local variables defined within recursive modules, the analyzer will stop capturing data if the number of recursive levels exceeds 128. A message is displayed in the trace list indicating that the recursion limit has been exceeded. Once the program returns to the 128th level of recursion, the capture of data is resumed. The count of recursive levels is relative to the start of the trace. If the trace was started in the middle of recursion (i.e., some recursive entries have occurred before the trace was started), it is possible for the analyzer to miss data. The analyzer will trace only recursive levels for which it finds entries. If the analyzer finds more exits than entries, a message is displayed in the trace list indicating that exits without matching entries have been detected.

It is more efficient to trace variables that are adjacent to each other in memory because the analyzer has only one range resource. The further apart in memory the variables are, the greater the time required to execute a trace. If the range of currently active variables includes variables within the monitor, the analyzer will halt the trace with an error message. The trace can be resumed with a smaller range specified without any loss of data. The trace must be halted since a break issued while in the monitor will be ignored. The monitor's variables are located where the monitor's static data has been linked to and on the stack below the currently executing procedure.

NOTE

The stack must be located at a lower address in memory than the static variables with the monitor **not** located between the two for static and local variables to be traced in a single measurement.

Command Syntax

The command syntax for setting up the trace variables measurement is shown in figure 6-7.

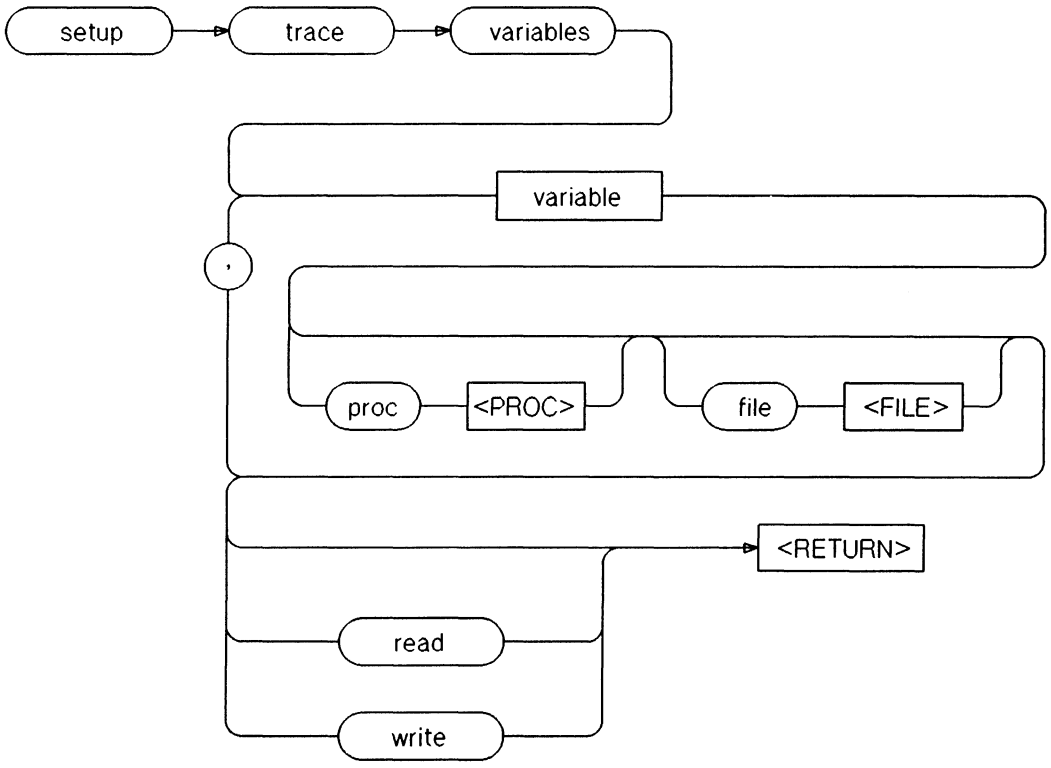
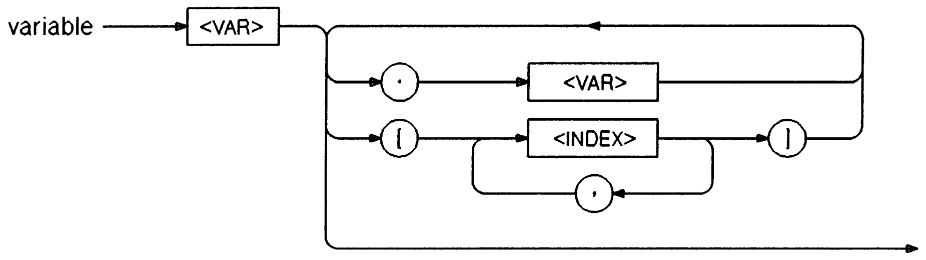


Figure 6-7. Setup Trace Variables Syntax Diagram

Parameters

The following paragraphs define the parameters used in the ***setup trace variables*** command.

<FILE>	<FILE> is an optional parameter that refers to the source file containing the specified <VAR> or <PROC> called out in the command statement. If the <VAR> or <PROC> is in the defined default path, the <FILE> parameter may be omitted from the command statement.
<INDEX>	Represents an index value (integer or scalar value) specifying a component of an array.
<PROC>	<PROC> is an optional parameter that refers to a procedure or function in which <VAR> is declared. If <PROC> is defined in the ***setup default_path*** command, it may be omitted in the ***setup trace variables*** command. If <PROC> is not specified in either the default path or the ***setup trace variables*** command, the analyzer assumes that <VAR> is a variable defined at the main program level.
read	***read*** specifies that only memory read accesses to the specified variable be traced. The default condition is to trace both memory read and memory write accesses to the specified variable.
<VAR>	<VAR> represents the name of a variable or parameter to be traced. <VAR> can be any valid Pascal or C variable expression. Pointer variables cannot be traced in the trace variables measurement mode.
variable	Variable may be any valid C or Pascal variable other than pointer types. Pointer variables cannot be traced with the trace variables measurement.



write	***write*** specifies that only memory write accesses to the variable be traced.

Examples

The following command examples illustrate how to use the ***setup trace variables*** command to define measurements.

setup trace variables pred_result ***proc*** proc2
setup trace variables COUNT , SNN ***proc*** PROC2 ,
    Q.FLAG ***file*** TESTP

The following example shows several lines of program, a setup trace variables command and a portion of the resulting trace list.

SOURCE PROGRAM LINES.

```
166     pred_result.enumerated = green
167     check = check + pred_result.arr[0]
168     check = check - result.arr[0];
169     check = check + pred-result.arr[1];
170     check = check - result.arr[1];
171     pred_result.enumerated = blue
172
173     u16();
174     pred_result.ch = 'a';
175     check = check + pred_result.arr[0];
176     check = check - result.arr[0];
177     check = check + pred_result.arr[1];
178     check = check - result.arr[1];
179     pred_result.ch = 'a';
```

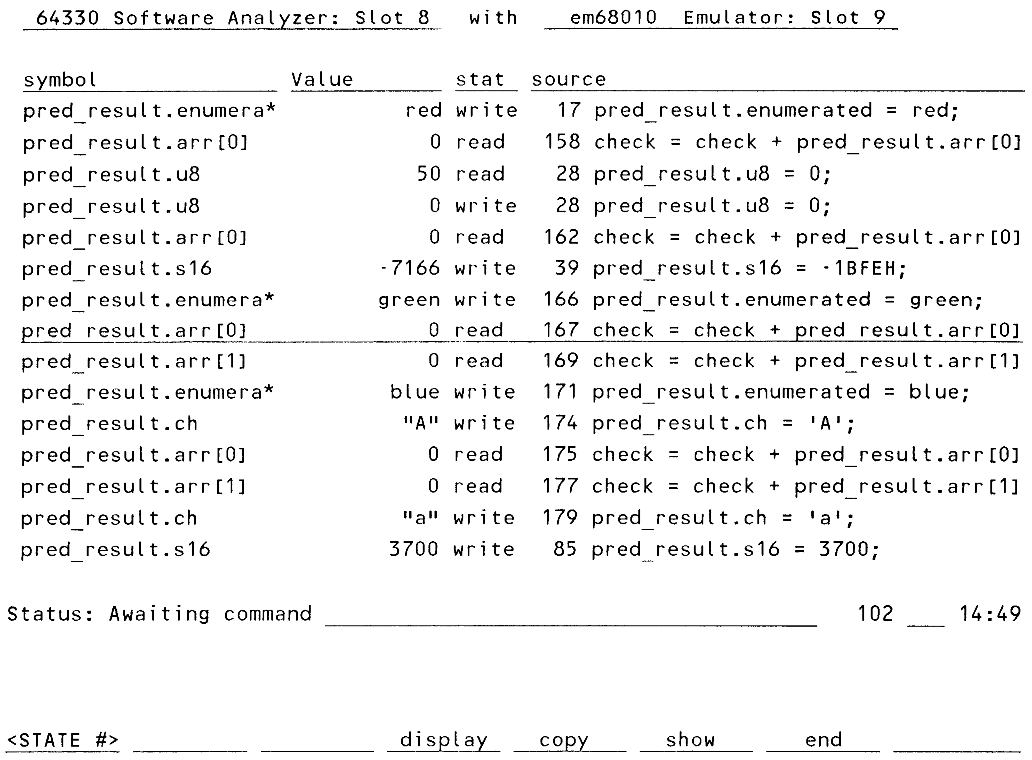
SETUP MEASUREMENT COMMAND.

setup trace variables pred_result

MEASUREMENT DISPLAY. Figure 6-8 is a trace variables measurement display. The trace list shows all accesses to the variable pred_result where pred_result is a structure. The value of the variable and the source line are shown.

The read and subsequent write of pred_result.u8 at source line 28 is due to the read then write nature of the instruction used by the target processor to clear a memory location.

pred_result.s16 is set to hexadecimal value -1BFE on line 39 in the source field but is displayed as decimal value -7166 in the value field. The default base for numeric data types is decimal.



```
 64330 Software Analyzer: Slot 8   with   em68010  Emulator: Slot 9

symbol                  Value        stat  source
pred_result.enumera*           red  write   17 pred_result.enumerated = red;
pred_result.arr[0]               0  read   158 check = check + pred_result.arr[0]
pred_result.u8                  50  read    28 pred_result.u8 = 0;
pred_result.u8                   0  write   28 pred_result.u8 = 0;
pred_result.arr[0]               0  read   162 check = check + pred_result.arr[0]
pred_result.s16              -7166  write   39 pred_result.s16 = -1BFEH;
pred_result.enumera*         green  write  166 pred_result.enumerated = green;
pred_result.arr[0]               0  read   167 check = check + pred_result.arr[0]
pred_result.arr[1]               0  read   169 check = check + pred_result.arr[1]
pred_result.enumera*          blue  write  171 pred_result.enumerated = blue;
pred_result.ch                 "A"  write  174 pred_result.ch = 'A';
pred_result.arr[0]               0  read   175 check = check + pred_result.arr[0]
pred_result.arr[1]               0  read   177 check = check + pred_result.arr[1]
pred_result.ch                 "a"  write  179 pred_result.ch = 'a';
pred_result.s16               3700  write   85 pred_result.s16 = 3700;

Status: Awaiting command                                    102     14:49


<STATE #>                    display   copy   show   end
```

Figure 6-8. Trace Variables Measurement Display

Chapter 7

USING INTERACTIVE COMMANDS FOR PROGRAM DEBUGGING

INTRODUCTION

The software analyzer has three commands that allow you to interact with the emulator without exiting the analyzer. These commands are ***setup breakpoints***, ***display*** variables, and ***modify*** variables.

The ***setup breakpoints*** command provides you with the capability to define software breakpoints. This enables you to command the emulation system to perform a "break on execution." The breakpoint may be specified as (1) any valid address of an opcode, (2) any Pascal or C source line containing executable code, or (3) the entry or exit point of a module (procedure or function). Up to 16 software breakpoints can be defined with the ***setup breakpoints*** command.

The ***display*** variables command displays the current value of a variable in emulation memory. The variable is specified in the same notation that it is specified in the source file. The ***modify*** variables command allows a variable to be modified manually by the user. The variable can be set to a specified value in integer format. The user program must be halted and the emulator running in the emulation monitor before the ***display*** or ***modify*** commands can be executed. Detailed descriptions of how to use these commands are given in this chapter.

If you have any difficulties or problems when using the software analyzer, see appendix E, Resolving Measurement Problems, for possible solutions.

BREAKPOINTS

The breakpoints feature of the software analyzer provides a method of positioning to the desired location in a users program to begin execution of a trace measurement. By using a series of breakpoints you can locate a position in the program under test to any combination of sequential address events. The breakpoint may be specified as (1) any valid address of an opcode, (2) any Pascal or C source line containing executable code, or (3) the entry and/or exit point of a module (procedure or function). The breakpoints are inserted at execution of a measurement and removed at measurement completion or halt. The instruction at the breakpoint address is always executed.

Command Syntax

The command syntax for setting up breakpoints is shown in figure 7-1.

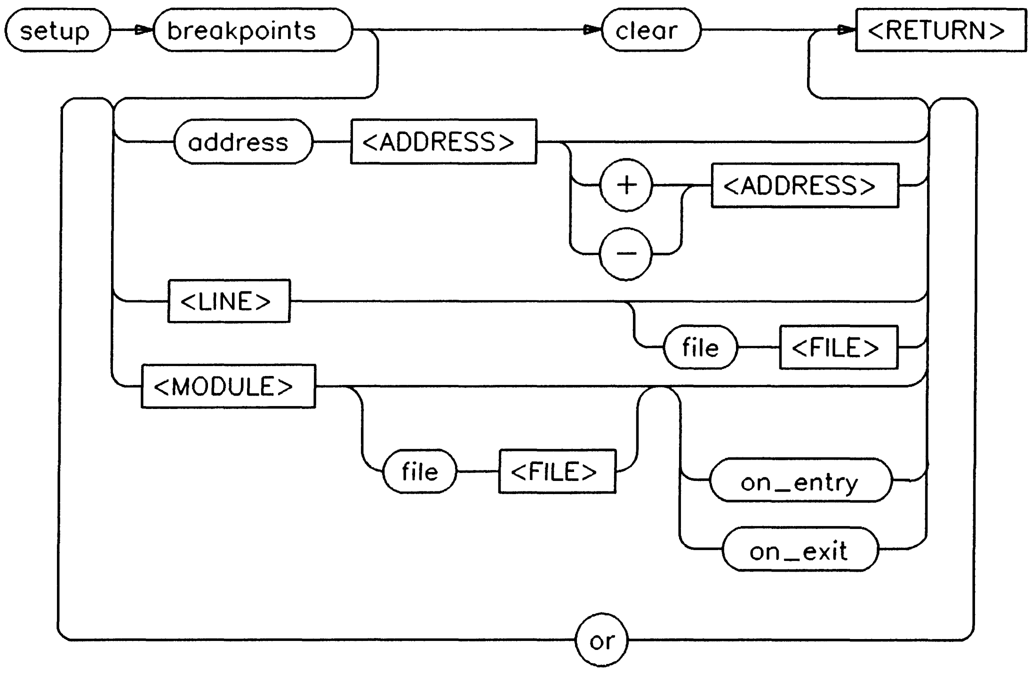


Figure 7-1. Setup Breakpoints Syntax Diagram

Parameters

The following definitions describe the parameters used in the ***setup breakpoints*** command.

<ADDRESS>	<ADDRESS> represents the address in the loaded absolute file at which a software breakpoint will be set.
clear	***clear*** causes all defined software breakpoints to be removed from the setup specification.

<FILE>	<FILE> is an optional parameter that refers to the source file containing the specified <MODULE> or line called out in the command statement. If the file containing the <MODULE> or line is the defined default path, the <FILE> parameter may be omitted from the command statement.
<LINE>	<LINE> represents the line number of a Pascal or C statement in the source program. If the specified <LINE> contains only comments (no executable code), the analyzer will associate the line number with the first line containing executable code following it. Any comment lines preceding the first line of executable code in a procedure or function are not recognized by the software analyzer.
<MODULE>	<MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure or a function within a specified file. In C, a module can be the name of a function within a specified file.
on_entry	***on_entry*** defines that the breakpoint be set at the entry point to the specified module only. The default condition is to set the breakpoint at both the entry and exit points of the specified module.
on_exit	***on_exit*** defines that the breakpoint be set at the exit point from the specified module.

Examples

The following command examples illustrate how to use the ***setup breakpoints*** command to define measurements.

setup breakpoints PROC2 ***on_entry***
setup breakpoints 102
setup breakpoints address 3068H + 20H
setup breakpoints 94 ***or*** BSORT ***on_exit or address*** 304BH

DISPLAY COMMAND

The *display* command displays the current value of a variable in emulation memory. The variable is displayed in the same notation that they were declared in the Pascal or C source file, i.e., in their declared data type format, i.e., as integers, reals, boolean values, characters, etc. The variable must be uniquely defined as to the module where it is declared. If the variable is defined outside of a module, i.e., a program variable in Pascal or outer level variable in C, then only the file name is required. The user program must be halted and the emulator running in the emulation monitor before the *display* command can be executed. The variable to be displayed must be accessible based upon the next address the program will execute. Local variables are accessible only if (1) the next program counter is within the procedure that defined the variable or (2), the variable belongs to a parent procedure of the current executing procedure.

Command Syntax

The command syntax for the *display* command is shown in figure 7-2.

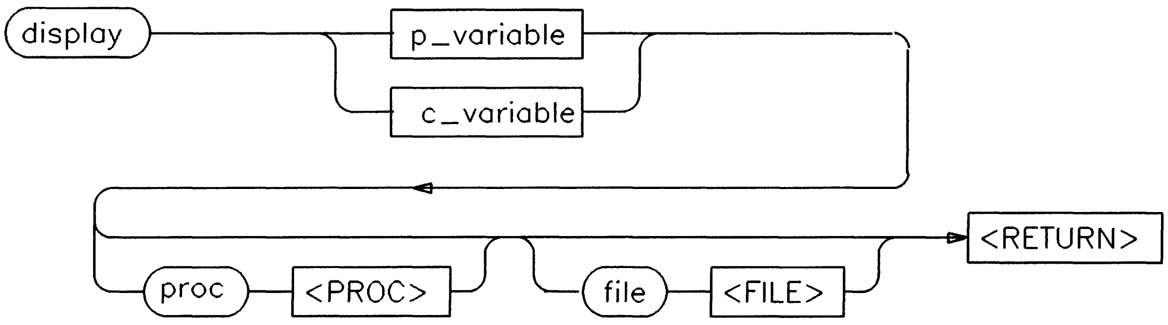
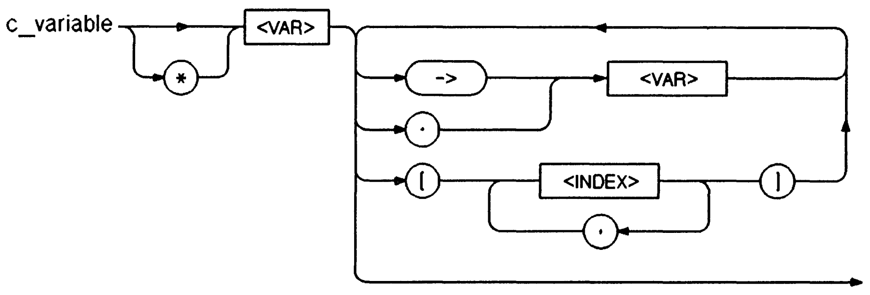


Figure 7-2. Display Variables Syntax Diagram

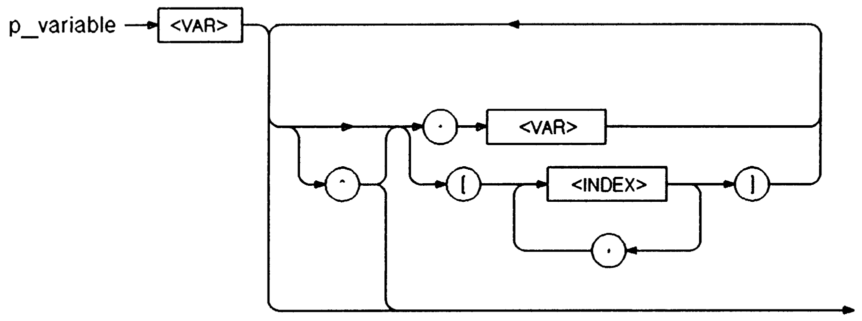
Parameters

The following paragraphs define the parameters used in the *display* variables command.

c_variable c_variable may be any valid C variable in the following expression format.



<FILE>	<FILE> is an optional parameter that refers to the source file containing the specified <VAR> and <PROC> called out in the command statement. If the <VAR> and <PROC> is in the defined default path, the <FILE> parameter may be omitted from the command statement.
<INDEX>	Represents an index value (integer or scalar value) specifying a component of an array.
<PROC>	<PROC> is an optional parameter that refers to a procedure or function in which <VAR> is declared. If <PROC> is defined in the ***setup default_path*** command, it may be omitted in the ***display*** command. If <PROC> is not specified in either the default path or the ***display*** command, the analyzer assumes that <VAR> is a variable defined at the main program level.
p_variable	p_variable may be any valid Pascal variable in the following expression format.



<VAR>	<VAR> represents the name of a variable or parameter to be displayed. <VAR> can be any valid Pascal or C variable expression.

Examples

The following command examples illustrate how to use the ***display*** command to display the value of variables.

display SNN^ ***proc*** PROC2 ***file*** NT1
display Q.FLAG ***proc*** CONTROLT
display A[1] ***file*** TESTP
display A^.B^.C^
display *a->b->c

MODIFY COMMAND

The *modify* command allows you to modify the current value of variables in emulation memory. Values must be specified in binary, octal, decimal, or hexadecimal notation. The variable must be uniquely defined as to the module where it is declared. If the variable is defined outside of a module, i.e., a program variable in Pascal or an outer variable in C, then only the file name is required. The user program must be halted and the emulator running in the emulation monitor before the *modify* command can be executed. The maximum variable size that can be modified with a single command is 32 bits. Larger variables must have their subelements modified individually with multiple commands. The variable must be accessible based upon the next address the program will execute. Local variables are accessible only if (1) the next program counter is within the procedure that defined the variable or (2), the variable belongs to a parent procedure of the current executing procedure.

Command Syntax

The command syntax for the *modify* command is shown in figure 7-3.

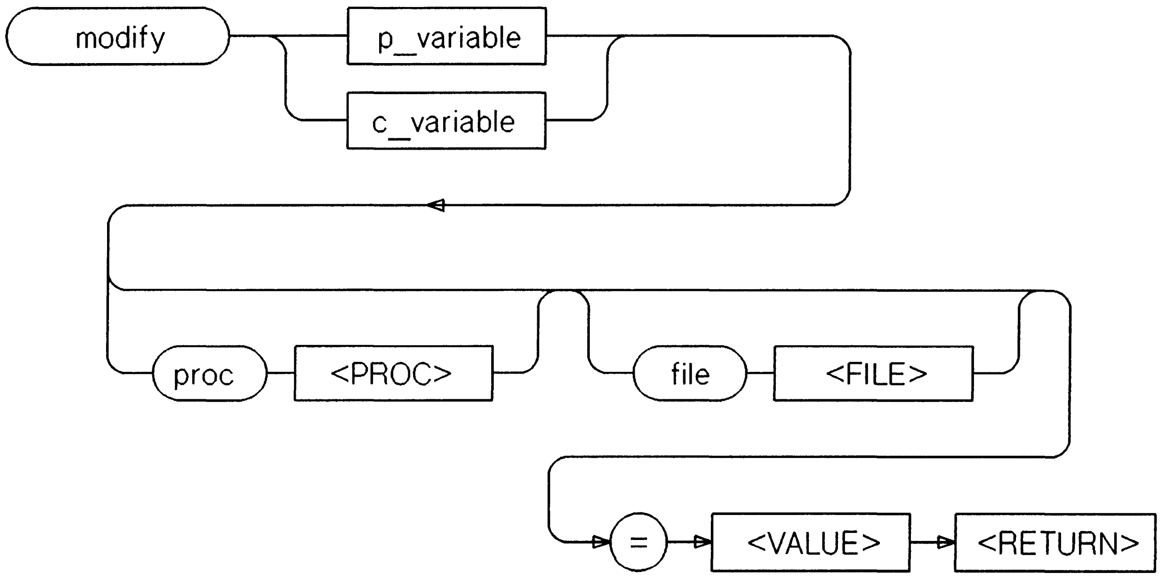


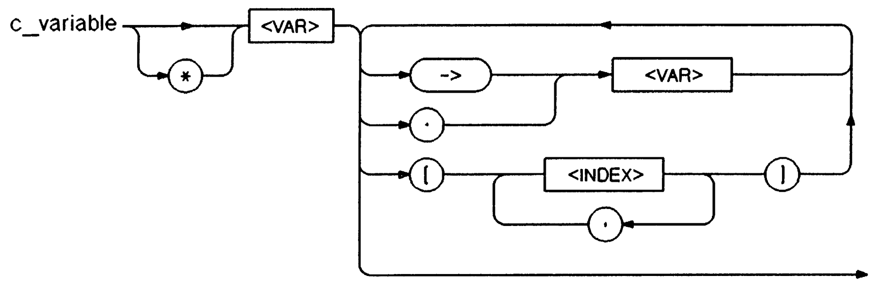
Figure 7-3. Modify Variables Syntax Diagram

PARAMETERS

The following paragraphs define the parameters used in the ***modify*** command.

c_variable

c_variable may be any valid C variable in the following expression format.



<FILE>

<FILE> is an optional parameter that refers to the source file containing the specified <VAR> and <PROC> called out in the command statement. If the <VAR> and <PROC> is in the defined default path, the <FILE> parameter may be omitted from the command statement.

<INDEX>

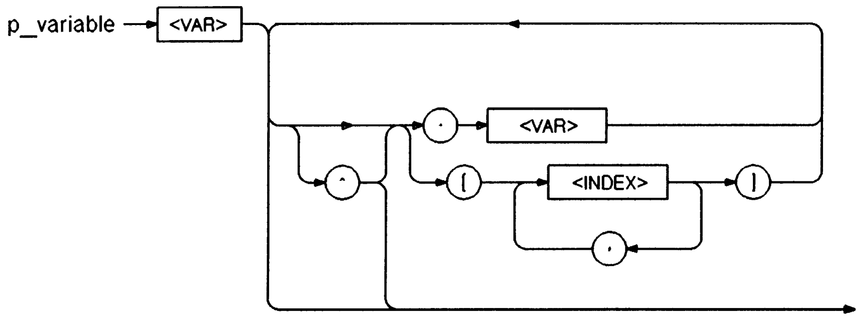
Represents an index value (integer or scalar value) specifying a component of an array.

<PROC>

<PROC> is an optional parameter that refers to a procedure or function in which <VAR> is declared. If <PROC> is defined in the ***setup default_path*** command, it may be omitted in the ***display*** command. If <PROC> is not specified in either the default path or the ***modify*** command, the analyzer assumes that <VAR> is a variable defined at the main program level.

p_variable

p_variable may be any valid Pascal variable in the following expression format.



<VALUE>

<VALUE> represents the value that the specified variable is to be changed to. <VALUE> must be specified as an integer value.

<VAR>

<VAR> represents the name of a variable or parameter to be modified. <VAR> can be any valid Pascal or C variable expression.

Examples

The following command examples illustrate how to use the ***modify*** command to change the value of program variables.

```
modify  Q.CHAR1  proc  LTRSORT = 41H
modify  NEXTINT = 0124H
modify  A^.B^.C^  =  15
modify  *a->b->c  =  15
```

Chapter 8

SELECTING AND FORMATTING THE MEASUREMENT DISPLAY

INTRODUCTION

This chapter describes how to select the data displayed in the measurement display and how to format the information to increase its usability. The measurement is automatically displayed in a trace list format upon completion of a measurement. If you have returned from the measurement display to the setup display, the measurement display may be recalled from the setup display by pressing the ***show*** softkey (***show_measurement*** command) followed by (RETURN). Four commands are available when the trace list is displayed; (1) ***display***, (2) ***copy***, (3) ***show***, and (4) ***end***. This chapter describes the ***display***, ***copy***, and ***show*** commands. The ***end*** command is described in chapter 10, Using Support Commands.

VIEWING THE MEASUREMENT DATA

The (ROLL UP), (ROLL DOWN), (NEXT PAGE), and (PREV PAGE) keys allow the user to scroll through the trace listing line-by-line or in page increments. The left and right cursor ((←) and (→)) keys used with the (SHIFT) key allow the user to move the display left or right on the screen. Pressing (RETURN) with no command left justifies the display. In addition, the user can enter an integer value (>= 0) that specifies the state in the trace data buffer to be centered on the display (highlighted in inverse video). If the state is not the first state in a display line, the software analyzer will align the display to the first state in the line.

DISPLAY FIELDS

The measurement display can consist of up to six different information fields; source, source_path, symbol, symbol_path, value, and status. The following paragraphs describe the information fields.

Source Field

The source field is made up of source file statements corresponding to line numbers contained in the asmb_sym file created when the file was compiled. The software analyzer compares addresses in the trace record with addresses associated with symbols in the analyzer database. If the analyzer detects an address corresponding to a line number symbol in the database, it extracts the source statement with its line number from the source file and stores it in the display buffer for display in the trace list. If a source line is not found, "????" is displayed in place of the line number and a message explaining the reason is displayed in the source field.

Source Path Field

The source_path field contains the file name and userid of the source file from which the source statement for the current display line was extracted.

Symbol Field

The symbol field contains the symbols traced by the software analyzer. In the trace modules measurement, the symbols are the names of the modules traced. In the trace data_flow measurement, the symbol field consists of a module name with the symbol field of subsequent lines containing the names of the parameters and variables being traced on entry to or exit from the named module. In the trace variables and trace statements measurements, the symbol field consists of variable and parameter names.

Symbol Path Field

The symbol_path field shows the path in which the symbol is defined. For modules, the symbol path contains a file name and userid. For variables and parameters, the symbol path may be a module name and file name with userid, or a file name and userid, depending upon the level at which the symbol is defined.

Value Field

The value field contains the values of the variables and parameters traced in the trace data_flow, trace statements, or trace variables measurements. This field is not valid in the trace modules measurement and cannot be displayed in the trace list display for that measurement. The value field shows the data values in the data type notation specified in the Pascal or C source program, i.e., integers are displayed as integer values, real numbers as mantissa and exponent portions, boolean values as true or false, etc.

Status Field

The status field indicates whether the traced operation is an entry or exit from a module if the traced symbol is a module name, or the traced operation is a read from or write to memory if the traced symbol is a variable or parameter.

INTERPRETING THE DISPLAY

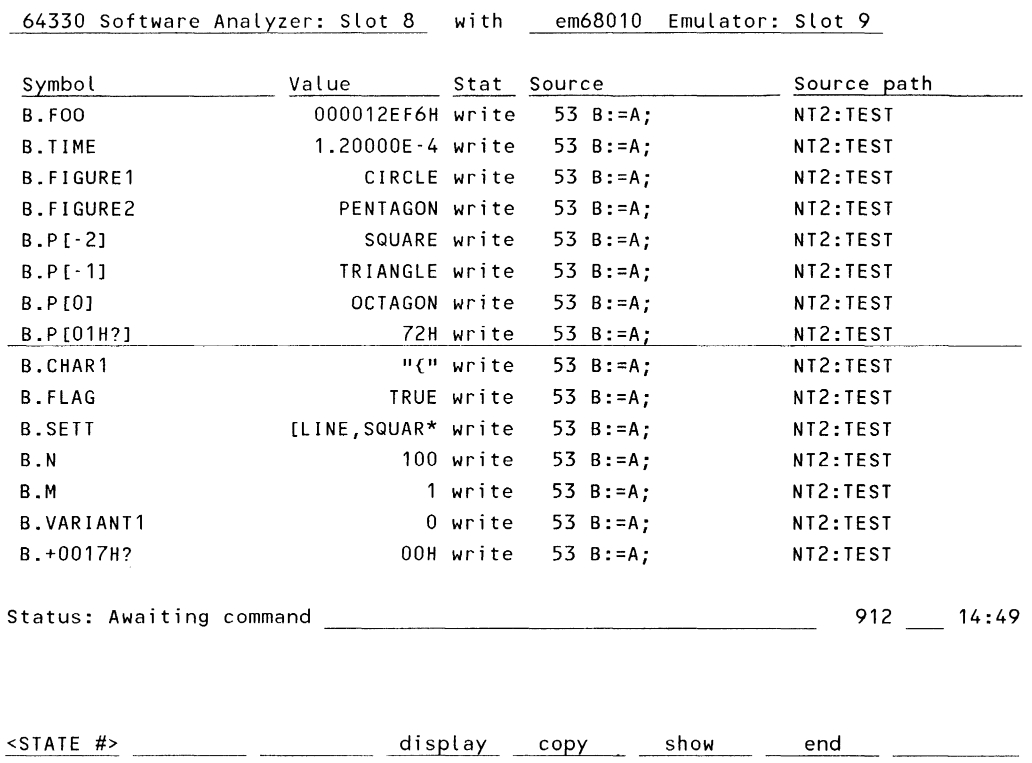
The following examples describe conventions and features of a trace list when displayed or copied to a list file. The examples were generated using the measurement command ***trace variables B file EXAMPLE***. Most of the discussion is applicable to displays generated by any software analyzer measurement. Part of the list file generated when file EXAMPLE was compiled is shown in figure 8-1. The TYPE and VAR definitions in the program must be understood before the sample displays can be understood.

```
  ...
  10 00000000  1  TYPE
  11 00000000  1  SHAPE          =(LINE,TRIANGLE,SQUARE,PENTAGON,HEXAGON,
  12 00000000  1                   HETAGON, OCTAGON, CIRCLE);
  13 00000000  1  POLYGON        =TRIANGLE..OCTAGON;
  14 00000000  1  SHAPE_SET      =SET OF SHAPE;
  15 00000000  1  POLY_ARRAY     =ARRAY[-2..0] OF POLYGON;
****WARNING ??                                      ^508
  16 00000000  1  PTR            =EC;
  17 00000000  1  REC            =RECORD
  18 00000000  2    FOO          :PTR;
  19 00000000  2    TIME         :REAL;
  20 00000000  2    FIGURE1      :SHAPE;
  21 00000000  2    FIGURE2      :SHAPE;
  22 00000000  2    P            :POLY_ARRAY;
  23 00000000  2    CHAR1        :CHAR;
  24 00000000  2    FLAG         :BOOLEAN;
  25 00000000  2    SETT         :SHAPE_SET;
  26 00000000  2    N            :UNSIGNED_16;
  27 00000000  2    CASE M       :SIGNED_16 OF
  28 00000000  2    2:(VARIANT2  :SIGNED_16);
  29 00000000  2    1:(VARIANT1  :SIGNED_8);
  30 00000000  1    END;
  31 00000000  1
  32 00000000  1  VAR
  33 00000000  1  A,B   :REC;
  ...
508: Warning: field or entry alignment; record or array comparisons may not work
```

Figure 8-1. Compiler Listing File For Program EXAMPLE

Current Line

In the trace list shown in figure 8-2, the underscored line in the center of the display is the current line. The number "912" on the status line is the first acquisition state for the current line. The number of states required for one line of display is variable.



```
 64330 Software Analyzer: Slot 8    with    em68010  Emulator: Slot 9

 Symbol                  Value      Stat  Source                 Source path
 B.FOO              000012EF6H write    53 B:=A;               NT2:TEST
 B.TIME              1.20000E-4 write    53 B:=A;               NT2:TEST
 B.FIGURE1               CIRCLE write    53 B:=A;               NT2:TEST
 B.FIGURE2             PENTAGON write    53 B:=A;               NT2:TEST
 B.P[-2]                 SQUARE write    53 B:=A;               NT2:TEST
 B.P[-1]               TRIANGLE write    53 B:=A;               NT2:TEST
 B.P[0]                 OCTAGON write    53 B:=A;               NT2:TEST
 B.P[01H?]                  72H write    53 B:=A;               NT2:TEST
 B.CHAR1                    "{" write    53 B:=A;               NT2:TEST
 B.FLAG                    TRUE write    53 B:=A;               NT2:TEST
 B.SETT           [LINE,SQUAR* write    53 B:=A;               NT2:TEST
 B.N                        100 write    53 B:=A;               NT2:TEST
 B.M                          1 write    53 B:=A;               NT2:TEST
 B.VARIANT1                   0 write    53 B:=A;               NT2:TEST
 B.+0017H?                  00H write    53 B:=A;               NT2:TEST

Status: Awaiting command                                        912     14:49


<STATE #>                        display   copy    show    end
```

Figure 8-2. Sample Display Showing How Pad Bytes, Variant Records, and Field Widths Are Displayed

Displaying Pad Bytes

Line 53 in source file EXAMPLE moves variable A in its entirety to variable B. The symbol field in the eight line of the display contains a question mark ("?"). A question mark in the symbol field indicates that a complete symbolic name does not exist for the value. The compiler warning in the type definition of POLY_ARRAY indicates that the physical size of a variable of type POLY_ARRAY is larger than the logical size. This is done to ensure that the subsequent field begins on an even byte boundary. The result is a "hole" or "pad byte" in variable B which has no symbolic name. The analyzer recognizes that the byte is physically part of array P but that an index of 1 is not valid. A question mark will always be displayed when the analyzer traces a pad byte corresponding to a "508" warning from the compiler.

Displaying Variant Records

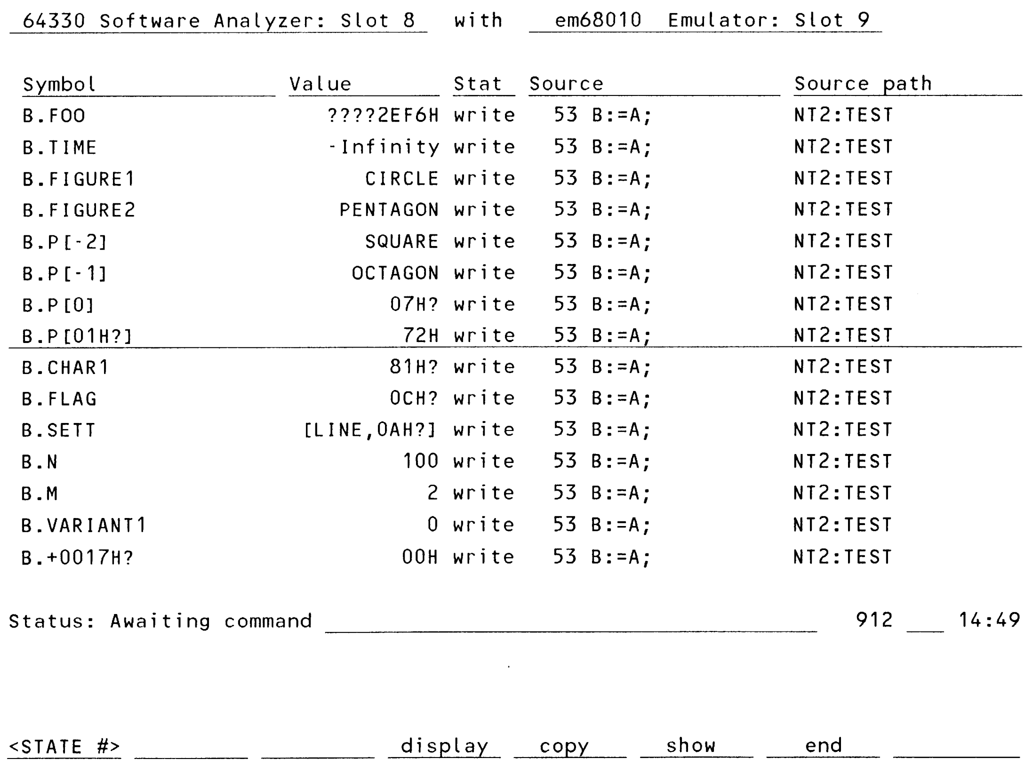
The question mark in the 15th line is slightly different. In Pascal, a record is physically large enough to accommodate its largest variant. In C, a structure is physically large enough to accommodate its largest union. In this case, the record B must be large enough to accommodate VARIANT2 which is two bytes long. Unless a specific variant is requested in the setup command, the analyzer defaults to displaying a record with the first variant (C language) or the last variant (Pascal language). The byte which is offset 17H bytes from the first byte of B is defined with respect to VARIANT2 but not with respect to VARIANT1. The analyzer always displays undefined bytes as +nnH?, where nn is a byte offset from the first byte of the record.

Field and Display Width

The "*" in the value field of the 11th line of the display indicates that the field is not large enough to display the entire value of B.SETT. The size of the field may be increased by using the display command to specify a larger width. The asterisk may appear as the last character in any field, indicating that there is additional information that is not being displayed. The display may be reformatted with a greater width for the field containing the asterisk. The analyzer limits the total width of the display to 132 characters. The screen is a 79 character window into this 132 character display. The window may be moved by using the SHIFT key with the left or right arrow.

Illegal Values

Figure 8-3 shows another display resulting from the trace variables B file EXAMPLE measurement. Question marks in the value field generally indicate that the value of a symbol is illegal with respect to the symbol type, i.e., it is out of range. The "07H?" in the seventh line is out of range because the value 7 (CIRCLE) is not a legal value for a POLYGON. In the ninth line, "81H" is not a legal ASCII character. In the 10th line, "0CH" is not a valid boolean value. The "0AH" in the 11th line indicates that the bit for the 10th element of B.SETT was set but there is no 10th element in the definition of the set (first element = element 0).



```
 64330 Software Analyzer: Slot 8    with    em68010  Emulator: Slot 9

 Symbol                     Value       Stat  Source                 Source path
 B.FOO                    ????2EF6H    write  53 B:=A;               NT2:TEST
 B.TIME                   -Infinity    write  53 B:=A;               NT2:TEST
 B.FIGURE1                   CIRCLE    write  53 B:=A;               NT2:TEST
 B.FIGURE2                 PENTAGON    write  53 B:=A;               NT2:TEST
 B.P[-2]                     SQUARE    write  53 B:=A;               NT2:TEST
 B.P[-1]                    OCTAGON    write  53 B:=A;               NT2:TEST
 B.P[0]                        07H?    write  53 B:=A;               NT2:TEST
 B.P[01H?]                      72H    write  53 B:=A;               NT2:TEST
 B.CHAR1                       81H?    write  53 B:=A;               NT2:TEST
 B.FLAG                        0CH?    write  53 B:=A;               NT2:TEST
 B.SETT                 [LINE,0AH?]    write  53 B:=A;               NT2:TEST
 B.N                            100    write  53 B:=A;               NT2:TEST
 B.M                              2    write  53 B:=A;               NT2:TEST
 B.VARIANT1                       0    write  53 B:=A;               NT2:TEST
 B.+0017H?                      00H    write  53 B:=A;               NT2:TEST

Status: Awaiting command _____________________________    912 ___ 14:49



<STATE #> ________ ________ display  copy  show  end  ________
```

Figure 8-3. Example Display Showing Illegal Values, Special Values, and Incomplete Access to Values.

Special Values

B.TIME in the second line of the display has a value of "-infinity". Real numbers may also have the special values "+infinity" and "not a number".

Incomplete Access To Variables

The four question marks in the value of B.FOO indicate that the analyzer has seen an incomplete access to the variable and has acquired only part of its value. In this example, the partial value is due to the analyzer beginning data capture in the middle of the two bus cycles that wrote to B.FOO. The asynchronous nature of the measurement resulted in the first write cycle not being captured.

Partial values also occur when the executing code accesses one traced variable and then a second traced variable without having accessed all of the first variable. A common example is when the measurement is tracing two 32-bit integers where one is assigned to the other. If the code does the assignment with word moves, partial values result from the second variable being written to before all of the first variable is read. Similarly, partial values are common when tracing two structured variables where one is assigned to the other.

Errors may occur in trace variables measurements when partial values are displayed adjacent to a complete value of the same value. The software analyzer groups bus cycles together into what it considers to be one logical access. This grouping may be incorrect when a access is adjacent to complete accesses. The user may alter the grouping by specifying a new integer position to be shown on the center line of the display. The software analyzer uses this integer position as the initial condition in its grouping algorithm. This will alter the manner in which the bus cycles are grouped.

DISPLAY COMMAND

The ***display*** command is used to specify the format of the information displayed in the trace listing. This command allows the user to select which fields to display, the sequence in which the fields are displayed, field width, and other parameters described below.

Command Syntax

The ***display*** command syntax is shown in figure 8-4.

Parameters

The following paragraphs describe the parameters used in the ***display*** command.

default	***default*** specifies that the trace list be displayed in the default format.
source	***source*** specifies that the source field is to be displayed in the trace list.
source_path	***source_path*** specifies that a field is to be displayed showing the source file name that the source statement was extracted from.
status	***status*** specifies that the status field is to be displayed in the trace list.

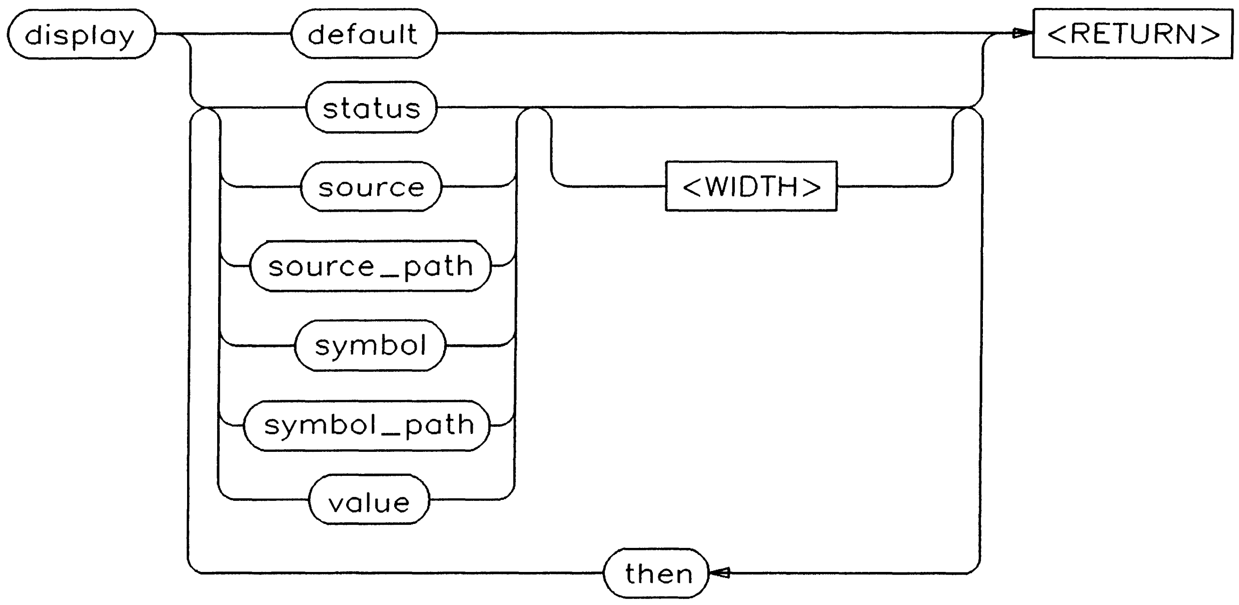


Figure 8-4. Display Command Syntax Diagram

symbol	***symbol*** specifies that the symbol field is to be displayed in the trace list.
symbol_path	***symbol_path*** specifies that a field is to be displayed showing the path in which the symbol is defined. For modules, the symbol_path contains a file name. For variables and parameters, the symbol_path may be a file or a module and file, depending upon the level at which the symbol is defined.
then	then is used as a delimiter to separate field definitions when more than one field is specified in the ***display*** command line.
value	***value*** specifies that the value field is to be displayed in the trace list. The values are displayed in the notation of the data type declared in the source program.
<WIDTH>	<WIDTH> is a prompt for the user to define the width of a displayed field in columns. Width must be an integer value from zero to 132 columns.

Examples

The following examples illustrate how to use the ***display*** command to format the trace list display.

display default
display source 40 ***then symbol*** 10 ***then value***
display symbol 10 ***then value*** 10 ***then source then source_path***

COPY COMMAND

The *copy* command is used to copy the current display or all or part of the measurement data (trace list) to a listing file or to the system printer.

Command Syntax

The *copy* command syntax is shown in figure 8-5.

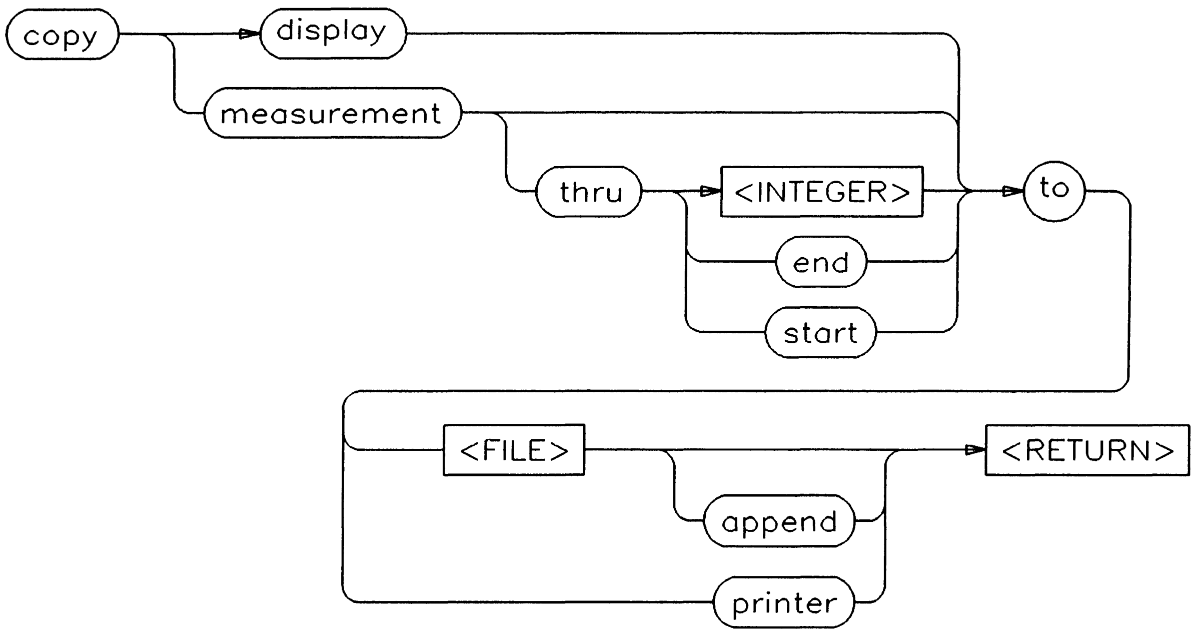


Figure 8-5. Copy Command Syntax Diagram

Parameters

The following definitions describe the parameters used in the *copy* command.

append	*append* specifies that the display or measurement being copied be appended to the end of the listing file.
display	*display* specifies that an image of the current display be copied to the specified file or system printer.
<FILE>	<FILE> is a prompt for the user to enter the file name of the listing file that the display or trace list is to be copied to.
measurement	*measurement* specifies that the measurement trace list or specified portion of the trace list be copied to the listing file or system printer.

printer	*printer* specifies that the display or trace list be copied to the system printer.
thru	*thru* is used to specify which portion of the trace listing is to be copied. *thru end* specifies the first line of the current display through the end of the trace list. *thru start* specifies the start of the trace list through the last line of the current display. <POSITION> specifies a record position in the trace measurement buffer. The trace list from the record position to the top or bottom of the current display will be copied, depending on whether the specified record position occurs after or before the currently displayed data. The minimum data that can be copied is the current display.

Examples

The following examples illustrate how to use the *copy* command.

copy display to printer
copy measurement to TRACE2
copy measurement thru 50 *to* TRACE1 *append*
copy measurement thru start to printer

When a *copy* command is executed, the acquisition state for each line of display or measurement trace list is copied along with the line.

SHOW COMMAND

The *show* (*show_setup*) command is used to exit the measurement display mode and return to the setup display.

Command Syntax

The syntax for the *show* command is shown in figure 8-6.

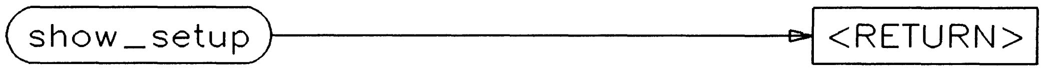


Figure 8-6. Show Command Syntax Diagram

STATE NUMBER PROMPT

<STATE #> is a prompt for the user to enter an integer value from 0 to 9999 specifying the position in the measurement buffer to be displayed and centered on the screen.

```
344 (RETURN)
```

Chapter 9

CONFIGURING THE ANALYZER

INTRODUCTION

This chapter explains how to configure the software analyzer. The analyzer can be configured manually each time it is used or it can be configured automatically. There are three methods that you can use to load the analyzer configuration automatically when you first enter the analyzer: (1) you can use the ***options continue*** feature to recall the measurement setup you used to perform the last tests, (2) you can use any measurement setup that has been previously stored in a configuration file, or (3) you can use a measurement system command file. Each of these methods are discussed in the following paragraphs.

NOTE

The software analyzer recomputes the addresses of symbols each time the analyzer is configured from a configuration file. If any programs referenced in a configuration file are changed such that symbol names are modified or deleted, the configuration file may no longer be valid. Errors will occur if symbols are no longer present or have been modified.

GETTING THE MEASUREMENT CONFIGURATION LAST USED

When you are running a measurement session and you press the ***end*** softkey the first time, the analyzer will automatically store the measurement configuration presently in use. You are now at the measurement system software level. You can enter the other analysis functions available at this level if you wish. To return to the software analyzer softkey level, press the ***sw_anly*** softkey, then the (RETURN) key.

Pressing the ***end*** softkey a second time brings you out of the measurement system level software to the system monitor level software. Here you can use the system monitor level softkey functions without disturbing the measurement setup you ended out of as long as you do not press the ***opt_test*** softkey at the monitor level. You can reenter the software analyzer measurement session with the last configuration used at any time by pressing the ***meas_sys*** and ***continue*** softkeys, and then the (RETURN) key.

NOTE

If you do not include the ***options continue*** statement in your command, your present measurement configuration will be purged.

The ***options continue*** function will not perform the function described above after the (RESET) key has been pressed twice, or after a power down or power fail, or after running performance verification. You can recover the configuration that was being used when you ended the last time by loading file Sw_anly<HP-IB address>:HP.

If you have entered the same emulator used by the analyzer after ending the analysis session and modified the emulation setup (i.e. loaded a new file while in emulation), an attempt to reenter the software analyzer using the ***options continue*** statement may fail.

GETTING A MEASUREMENT CONFIGURATION FROM A CONFIGURATION FILE

The analyzer can store complete measurement configurations in disc memory so that you can keep a library of test setups and measurement data on hand for your measurement needs. You can then load a selected measurement configuration to suit your current need without having to build a new configuration for each measurement session. If you have a configuration file that is close to the configuration you need, you can load it and then modify it, saving time by eliminating the requirement to enter some basic parameters. The following paragraphs describe the procedures used to store and recover these measurement configurations. The syntax for saving or loading a configuration is given in Appendix A.

Saving A Measurement Configuration

1. Set up any desired measurement configuration in your software analyzer. It is a good idea to set up a good basic configuration that can be stored, then loaded and used as a building block for other measurement configurations.

2. Press the ***configure*** and ***save_in*** softkeys, then type in an A in answer to the ***<FILE>*** softkey prompt. Now press the (RETURN) key. The analyzer will now save its present measurement configuration in the trace file you have just named A. The file will be stored under the current USERID.

3. Now you can change the setup any way you like. Your original measurement configuration will still be saved exactly as you stored it in file A. You can use this procedure to make as many configurations as you may require. These, in turn, can also be stored in configuration files for access at a later time.

If you used the ***write_protected*** option when you saved your configuration and you ever want to purge that file, you must return to the system monitor level software to accomplish the purge. To accomplish this press the ***end*** softkey, then the (RETURN) key (in that sequence) two times. This will return you to the system monitor level software. You can now purge the unwanted file.

NOTE

When you reenter the software analyzer, you can recover the the measurement data acquired in the last measurement made before ending the previous measurement session. The data is stored in file Swtrace<analysis slot #><HP-IB address>:<userid>:trace. During a measurement session this is the data file for the current measurement. A ***show measurement*** command will always display the contents of this file.

Loading A Measurement Configuration

If you are starting a session (or are at the measurement system monitor level) and want to load a configuration you have previously stored in a file, proceed as follows: press the ***sw_anly*** softkey, type in the name of the configuration file you want to use, and press the RETURN key. You will gain access to the software analyzer and it will automatically search the disc and load the configuration you stored in the file you requested.

If you are operating the software analyzer in a measurement session and you want to load a configuration you have stored in a file without ending out of the analyzer, proceed as follows: press the ***configure*** and ***load_from*** softkeys, type in the name of the file you want to use, and press the RETURN key. This will cause the analyzer to purge the present measurement setup and load the configuration from the file you requested.

GETTING A MEASUREMENT CONFIGURATION FROM A COMMAND FILE

The command file must contain the sequence of command lines required to create the setup from the monitor level. Using the parameter passing feature of command files will allow the emul_cmd, absolute, and source files required by the software analyzer to be specified in the command file. An example is shown below:

```
PARMS &CMD_FILE &ABS_FILE &SOURCE_FILE
measurement_system
sw_anly
&CMD_FILE
load &ABS_FILE
setup default_path &SOURCE_FILE
setup trace_records 100
setup trace statements line 20 thru 100
run at_execution from transfer_address
execute

Where:
    CMD_FILE is the emulation command file
    ABS_FILE is the linked absolute file to be traced
    SOURCE_FILE is the source file to be debugged
```

The command file can be run from the system monitor level. To run the command file, type in the file name and the file names for the emulation command file, absolute file, and source files to be used.

<CMDFILE> STEP2 NT1 NT1 (RETURN)

If the file names are not entered, the prompts:

Define parameter &CMD_FILE:
Define parameter &ABS_FILE:
Define parameter &SOURCE_FILE:

will appear on the command line.

Chapter 10

USING SUPPORT COMMANDS

INTRODUCTION

This chapter describes the software conventions used to make keyboard entries for operating the software analyzer and how the analyzer directs the entries you make. Also described are the utility softkeys, the utility keyboard keys, and the prompt softkeys.

SYSTEM SOFTWARE CONVENTIONS

This section contains information concerning the system software as it relates to any of the subsystems installed in a particular mainframe.

User Identification

The user identification (userid) command is the means of identifying yourself to the HP 64000 system software as a unique individual who will be using the system for your own analysis/development projects. Signing onto the system with your own userid immediately identifies which group of files the system is to work with.

The ***userid*** syntax is a string of up to six (6) alphanumeric characters which start with an upper case alpha character. If you select a userid with more than six characters, the system will recognize only the first six. If you do not select a userid, the default condition is a blank userid. A blank userid limits your ability to designate a file because if more than one file is given the same name, and that file is called up, the system will recognize the first one it sees (which may or may not be the one you want).

Directed Syntax

The system software causes a row of softkey labels to be displayed across the bottom line of the CRT display. These softkey labels identify the functions to be obtained by pressing corresponding keys in the row at the top of the keyboard. When you press one of the softkeys (selecting a parameter), the names of all the softkey labels change. The new softkey names offer selections that can be made to complete the command entry.

By directing the syntax of your entries, syntactical errors are virtually eliminated. The softkey label line always identifies appropriate entries to be made at any point during the process of formulating a command. The software analyzer softkeys always prompt the user with a <***RETURN***> softkey label when a valid command statement has been entered. If the softkey label line contains more labels than the <***RETURN***> softkey prompt, then the command statement may either continue or be terminated by pressing the RETURN key, as determined by the specific requirement of the command being formulated.

Entering Numeric Values

You can enter numbers into an analysis specification in any of the four standard number bases. Place the applicable letter symbol (B, O or Q, D, H) at the end of your number to define its base. Refer to the following examples:

1000B = 1000 binary
1000O or 1000Q = 1000 octal
1000H = 1000 hexadecimal
1000D or 1000 = 1000 decimal

Hexadecimal numbers beginning with a letter must be preceded with a numeric zero. For example:

3FAH, 0FFH, 0F44H (but not F44H)

NOTE

Decimal is assumed if no base is specified when a number is entered.

Entering Module/Variable Names

You can enter module and variable names into an analysis specification exactly as it appears in the source program with one exception. If the module or variable name is lower case and is identical to a software analyzer keyword, it must be specified in quotes, e.g., "entry". Otherwise the analyzer interprets the name as a keyword and generates an error message.

Command Files

A command file is a source file containing a sequence of commands as they would appear on the command line if entered manually from the softkeys or keyboard. A command file is used to create a particular measurement configuration programatically. A command file provides a self-documenting record of a measurement setup and allows easy editing and modification. By using the ***wait*** command described in this chapter, command files can be set up to perform some automated measurements which require no operator interaction.

A semicolon (;) is used in the command file to denote comments. The analyzer software will not read any material following a ";" in any line of a command file. It will start loading new instructions only after it finds the next carriage return.

In the example:

```
run  from  transfer_address; causes program execution to begin
```

only the command line text "**run from transfer_address**" will be acted on.

Logging Commands

The HP 64000 Logic Development System has the capability to log commands to a command file. This feature is especially useful for building command files that will carry out the entire measurement setup automatically. To log commands for a measurement setup session from the system monitor level software, press the *log* and *to* softkeys, type in the name of the file you want to use, and press the RETURN key. From this point until you are once again back in the system monitor software and press the *log* and *off* softkeys, and the RETURN key, all of the valid commands you entered are logged into the log file. You may then conduct a software analysis session which will build a command file for later use or for modification.

File Names

File names may consist of from one up to nine alphanumeric characters, starting with an upper case letter. Underscores (_) are also permissible. Alpha characters, after the first character, may be upper or lower case.

SOFTKEY UTILITY COMMANDS

The softkey utility commands available in the software analyzer are the *execute, copy, show, end, halt,* and *wait* softkeys. These softkeys allow the user to execute a measurement, copy and display information concerning the current session, and to end the analysis session without losing the current measurement specification. In addition they allow the user to halt a measurement in progress or to disable commands while a measurement is in progress. The utility softkey commands are described in detail below.

Execute Softkey

The *execute* softkey causes the analyzer to initiate a measurement based on the parameters defined in the measurement setup specification. Pressing the *execute* softkey, then the RETURN key causes the analyzer to search for and acquire data as specified in the trace setup specification. While the measurement is in progress, the STATUS line will read "**Executing, number of states = n**". When the specified number of trace records have been acquired, the STATUS line will read "**M68010: Running in monitor trace complete**" followed by "**formatting display**" while the data is being formatted for display. When formatting is completed, the measurement will appear on the screen. Whenever the *execute* command is given, the last measurement file is deleted.

Copy Softkey

The *copy* softkey allows you to copy the current measurement setup parameters, the current measurement, or the current display to a file that you name or to the system printer. The information may also be appended to an existing listing file.

Show Softkey

The *show* softkey allows you to alternately select the current measurement data or the current measurement setup for display on the screen. To view all the data available, use the ROLL UP, (ROLL DOWN), (NEXT PAGE), and (PREV PAGE) keys, and the shifted left and right cursor keys ((SHIFT)(←) and (SHIFT)(→)).

End Softkey

The *end* softkey is used for transportation from the present software level to the next higher level. When you press the *end* softkey, the system will exit the software analyzer and return to the measurement system level software. When you leave the analyzer by use of the *end* softkey, the measurement configuration will be stored on the disc. Pressing the *end* softkey a second time causes the software to enter the system monitor level of software. You can make use of the system monitor level softkey functions without disturbing the measurement you ended from as long as you do not press the *opt_test* softkey at the monitor level. If you pressed the *end* softkey only once, the instrument is at the measurement system software level. From here you can reenter the analysis module by pressing the *sw_anly* softkey and the (RETURN) key. If you pressed the *end* softkey twice the instrument is at the system monitor software level. From here you can reenter the analysis module by pressing the *meas_sys* and *continue* softkeys and the (RETURN) key.

NOTE

If you do not include the ***options continue*** statement in your command to return from the system monitor level software, your present measurement configuration will be purged. The measurement configuration is still available in the default configuration file Sw_anly<HP-IB>:HP where <HP-IB> is the HP-IB address of the HP 64000 station.

Halt Softkey

The *halt* softkey is used to halt execution of the current measurement. The halt function is accomplished by pressing the *halt* softkey, then the (RETURN) key. The halt command also executes a break and puts the program into the monitor.

Wait Softkey

The *wait measurement_complete* command causes the software analyzer to disable all software analyzer commands until the measurement is completed. When used in a command file, the *wait measurement_complete* command suspends execution of the command file until the measurement is completed. This enables the user to create command files that can execute repetitive measurements, storing the measurement results between measurement executions. This command provides the capability to automatically make measurements, unattended by the user, with the results stored in listing files for future analysis.

KEYBOARD UTILITY KEYS

The following keyboard keys are used to help you to make entries in the software performance analyzer command lines shown on the display.

Recall Key

The (RECALL) key will cause the analysis module to return the preceding valid command line to the screen. The analysis module has a command line memory which the (RECALL) key accesses. Each time you press the (RECALL) key, the analyzer steps one execution further back into its memory of command lines.

Tab Key

The (TAB) key is used to move the cursor rapidly through the command line on screen. This key is useful when you are making modifications to long specifications. By pressing (TAB), you step the cursor from entry to entry forward through the specification on the command line. By pressing the (SHIFT) key and then the (TAB) key, you step the cursor backwards through the specification.

Insert Char And Delete Char Keys

The (INSERT CHAR) and (DELETE CHAR) keys are used to edit the content of the command line. The (INSERT CHAR) key will open a space before the present position of the cursor so that you can add entries in the command line. The remainder of the line will automatically shift to the right with each new entry that you make. The (INSERT CHAR) key function will remain in effect until it is pressed again or until any other utility key is pressed (except (←), (→), or (CAPS LOCK)). The (DELETE CHAR) key is used to eliminate entries from the the command line without losing the entire specification. When you press the (DELETE CHAR) key, the entry directly over the cursor will be eliminated and the remainder of the specification will shift left. Holding the (DELETE CHAR) key down will cause multiple character deletions as characters are shifted left, over the cursor position.

PROMPT SOFTKEYS

Any softkey name enclosed in angle-brackets "<>" is a prompt for the operator. If you press a prompt softkey, the STATUS line of the display will explain the meaning of the prompt. The software analyzer softkey label prompts and their corresponding status line prompt messages are given in appendix B.

NOTES

Chapter 11

SYMBOLS AND DATA TYPES

INTRODUCTION

This chapter describes the storage classes and data types that the software analyzer recognizes. Symbols are categorized by whether the location of the symbol is known at link time (static) or whether the location changes during run time (dynamic). The scalar and structured data types that can be symbolically referenced by the software analyzer are described.

SYMBOL CLASSIFICATIONS

Static Symbols

LOCAL AND GLOBAL VARIABLES. Static variables (both local and global) are defined as those whose base locations are allocated at link time. The software analyzer expects that these locations will not change during run time.

PROGRAMS, MODULES, PROCEDURES, AND FUNCTIONS. A module is a set of program statements that can be invoked (or referred to) by name. In Pascal, "module" may refer to the main program of a file, or to individual procedures or functions within a program. In C, "module" can refer only to functions. The key elements of a module, as required by the analyzer, are (1) the module must be a contiguous segment of code with a single entry point and a single exit point, and (2) all the code for the module must fall within the range of the entry and exit points. The entry point is defined in terms of the assembly code which is generated after compilation of the high level language module. It is the first executable instruction of the module (this includes any compiler overhead which may have to be done before the assembly code performing the actual module operations begins). The address of the entry instruction becomes the lower boundary of the address range for the module. The exit point is defined as the last executable instruction of the code segment and its address becomes the upper boundary of the address range for the module.

In the commands ***setup trace data_flow <MODULE> ...***, and ***setup breakpoints <MODULE> ...***, the user can specify where a break should occur, either on entry to the module or upon exit from the module. If neither ***entry*** nor ***exit*** are specified, the break is set upon both the entry and exit points of the module. The command ***setup trace modules ...*** causes a break on both the entry and exit points within the modules included in the measurement. In the command ***setup trace statements <MODULE> ...*** , an address range is set up such that all statements within the module cause a break to occur.

In the measurement commands ***setup breakpoints ..., setup trace statements...*** and ***run ... from ...***, either a module name or a line number may be specified as the starting point. If a module name is given, the measurement will begin at the entry point into the module (code including compiler overhead). To avoid starting with the compiler overhead code, the first source code line number of the module may be given to identify the module itself instead of the module name. Then, the asmb_sym file can be consulted to find the association between the line number and the assembly code it relates to and the first instruction of the assembly code reflecting the actual module instructions will become the starting point for the measurement.

LABELS. The entry point into a module's assembly code must have a label associated with it which must be the module name (e.g. MAIN). The exit point must also have a label associated with it which is identical in the first 14 characters to the entry point label except that an "R" is appended to the front of the label (e.g. RMAIN). It is these labels, found in the symbol files, that the analyzer keys on to perform a table lookup of the address range associated with the module, as well as its entry and exit points. The compilers follow these design rules but may, under certain conditions, create identical labels. These conditions are as follows:

1. procedures and functions in Pascal which are on different levels (i.e. nested procedures) may have identical names.

2. Due to the creation of the "R" or exit point labels, procedures and/or functions identical in the first 14 characters on any level will produce identical "R" labels.

The analyzer always keys on the first label it encounters that matches the specified label. Therefore, in order to avoid having the analyzer key on the wrong label, it is recommended that you always make your procedure and/or function names unique within the first 14 characters.

LINE NUMBERS. The software analyzer provides symbolic lookup of line numbers. These line numbers correspond to the line numbers found in the compiled listing file. The analyzer only accepts line numbers having executable code associated with them. If a line number has several instructions associated with it, the first instruction is the instruction associated with that line number. Lines that are intermixed in high level code, but contain only comments, do have executable code associated with them. These lines will be associated with the executable code immediately following them. Any comments that occur before the beginning and after the end of a module do not have executable code associated with them and do not exist for purposes of the software analyzer.

PATHS. A path consists of a module name and source file name that uniquely identifies a variable. Possible module names include function names in C programs, and procedure and function names in Pascal programs. The procedure or function name may be qualified by a file name. In Pascal, the main program path is defined by the file name.

Proc. The keyword ***proc*** is found in the commands ***modify <VAR> proc ...***, ***display <VAR> proc ...***, ***setup trace variables <VAR> proc ...***, and ***setup trace data_flow ...***. The keyword ***proc*** is used to specify an element of the variable's path, i.e., ***proc*** defines the procedure or function the variable belongs to and enables unique identification of the variable.

File. The keyword ***file*** is used to describe the file a variable belongs to. If only a file is given with no ***proc*** specified, the variable belongs to the outermost level by default. In Pascal, the outermost level is the program level.

Default Path. A default path may be set up before an actual measurement command is given. Then, if no path name is specified in the measurement command, the default path is used as the path definition for the measurement. If no default path is defined, the path must be specified in the measurement command.

Dynamic Symbols

LOCAL VARIABLES. Dynamically activated local variables are those that are assigned to the stack when the procedure is invoked and taken off when the procedure ends. The variable can be in different places at different times during run time.

REFERENCE PARAMETERS. A parameter that is passed by reference causes the address of the parameter to be passed. If a variable passed by reference is traced, the changes that occur to that variable will be seen within the subroutine the parameter was passed to. The variable will also be displayed with the name it acquired after being passed as a parameter.

VALUE PARAMETERS. Value parameters are not active on exit from procedures since they are treated in the same manner as local variables by the compiler. In C, all parameters except arrays are passed by value.

SYMBOLIC DATA TYPES

Many types of data can be symbolically referenced by the software analyzer. The following paragraphs describe the data types recognized by the software analyzer.

Intrinsic Data Types

Table 11-1 describes the intrinsic data types recognized by the software analyzer.

Table 11-1. Intrinsic Data Types

Scalar Data Types

Pascal	C	Description
BOOLEAN	Not applicable	An 8-bit value whose low-order bit represents the value TRUE (1) or FALSE (0).
BYTE	short	An 8-bit signed integer in the range -128 to +127.
SIGNED_8	short	An 8-bit signed integer in the range -128 to +127.

Table 11-1. Intrinsic Data Types (Cont'd)

Scalar Data Types (Cont'd)

Pascal	C	Description
UNSIGNED_8	unsigned short	An 8-bit unsigned integer in the range 0 to 255.
SIGNED_16	int	A 16-bit signed integer in the range -32768 to +32767.
UNSIGNED_16	unsigned	A 16-bit unsigned integer in the range 0 to 65535.
SIGNED_32	long	A 32-bit signed integer in the range -2,147,483,648 to +2,147,483,647.
UNSIGNED_32	unsigned long	A 32-bit unsigned integer in the range 0 to +4,294,967,295.
CHAR	char	An 8-bit value in the set of characters defined by the 8-bit ASCII character set.
INTEGER	long	A 32-bit signed integer in the range -2,147,483,648 to +2,147,483,647.
REAL	float	A 32-bit binary value representing a floating point number in IEEE simple precision format.
LONGREAL	double	A 64-bit binary value representing a floating-point number in IEEE double precision format.

User-Definable Data Types

Pascal	C	Description
SCALAR TYPE	enum	A type that defines an ordered set of values by enumerating the identifiers which denote these values.
SUBRANGE TYPE	Not Applicable	A type that is identified as a subrange of a previously defined ordinal type (char, byte, integer, or scalar) in which the smallest and largest values are user defined.

Structured Data Types

The following paragraphs describe the structured data types recognized by the software analyzer.

ARRAY. An array is a structure consisting of a number of components which are all of the same type (called the component type), in which the components (elements of the array) are accessed by index expressions. The array type definition specifies the component type and, in Pascal, the index type.

In Pascal, the component type may be of any type. Multidimensional arrays may be represented as "ARRAY OF ARRAY (OF ARRAY..)" with an arbitrary number of indices. The index type must be a simple type such as scalar or subrange type.

In C, the component type may also be of any type. Multiple dimensions may be specified by multiple brackets, i.e., [size] [size] [size]. The index type must always be integer. In C, an array passed as a parameter with undefined size cannot be traced with the software analyzer.

POINTER. A pointer is a variable that contains the address of a dynamic variable such that the dynamic variable can be accessed via the pointer variable. In C, arrays are not considered pointers for the purposes of this manual. Pointers can be traced in all software analyzer measurements that trace variables. Pointer expressions (the dynamic variable accessed via the pointer variable) can be traced only with the ***trace data_flow*** measurement. Pointer expressions can be displayed or modified using the ***modify*** and ***display*** commands.

SET (Not Applicable to C). A set is a structure defining the set of values that is the power set of its base type,(i.e., the set of all subsets of values of the base type). The base type must be a scalar or subrange type.

RECORD/STRUCTURE. A record (structure in C) is a data type consisting of a fixed number of components, called fields (members in C), each of which can be of any type. For each field/member, the record/structure definition specifies a field/member name identifier and the field/member type. For the remainder of this discussion, the Pascal terminology will be used.

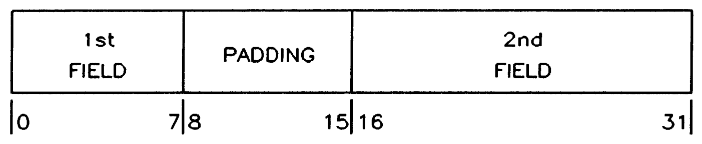
VARIANT RECORDS/UNIONS. When the Pascal or C compiler allocates space for variant record fields, every field in the variant section is allocated the amount of space necessary to accommodate the largest variant field. Padding is used to fill up unused space for those variant fields which are smaller than the largest variant field.

If a variant field is to be displayed, the specific name of the variant field should be specified in the measurement. If it is not (i.e. the record as a whole is specified), the analyzer will display the record in terms of the first (C source code) or last (Pascal source code) variant field. Therefore, padded space could be displayed as meaningful contents or vice versa.

Using the following program example, tracing AREC.SECOND would display a 32-bit value. However, if AREC was traced, the display would show the 16-bit field in AREC followed by 48 bits shown as padded space.

```
EXAMPLE:  (Pascal)
  TYPE
    VREC = RECORD
             CASE TAG : INTEGER OF
                 1:  (FIRST: A_16_BIT_SIZE);
                 2:  (SECOND: A_32_BIT_SIZE);
                 3:  (THIRD: A_64_BIT_SIZE);
           END;
  VAR
    AREC : VREC;
    VAR1 : A_16_BIT_SIZE;
    VAR2 : A_32_BIT_SIZE;
    VAR3 : A_64_BIT_SIZE;

  BEGIN
   :
   .
  CASE AREC.TAG OF
      1:  AREC.FIRST := VAR1;
      2:  AREC.SECOND := VAR2;
      3:  AREC.THIRD := VAR3;
  END;
   :
   .
  END;
```

During compilation, padding may take place automatically to handle memory alignment. For example, in a record defined as having an eight-bit field followed by a 16-bit field, the eight bits between the two fields may be padded to accommodate word boundaries.



If the entire record was displayed, it would contain an 8-bit field of padding. See chapter 8 for an example of how pad bytes are displayed.

Appendix A

OPERATING SYNTAX DIAGRAMS

INTRODUCTION

This appendix contains the operating syntax diagrams for the software analyzer. These diagrams are based on the guided-syntax softkeys that appear when the software analyzer is being used. The following syntax diagrams are provided in this appendix.

Figure No.	Description	Page No.
A-1.	Software Analyzer Level Syntax Diagram	A-2
A-2.	Run Syntax Diagram	A-3
A-3.	Setup Syntax Diagram	A-3
A-4.	Setup Trace Data_Flow Syntax Diagram	A-4
A-5.	Setup Trace Modules Syntax Diagram	A-4
A-6.	Setup Trace Statements Syntax Diagram	A-5
A-7.	Setup Trace Variables Syntax Diagram	A-5
A-8.	Setup Breakpoints Syntax Diagram	A-6
A-9.	Setup Default_Path Syntax Diagram	A-6
A-10.	Database_check Syntax Diagram	A-6
A-11.	Display Variables Syntax Diagram	A-7
A-12.	Modify Variables Syntax Diagram	A-7
A-13.	Show Command Syntax Diagram	A-7
A-14.	Display Command Syntax Diagram	A-8
A-15.	Copy Command Syntax Diagram (Measurement Display)	A-8
A-16.	Load Command Syntax Diagram	A-9
A-17.	Configuration Command Syntax Diagram	A-9
A-18.	Copy Command Syntax Diagram (Setup Display)	A-9
A-19.	Variable Syntax Diagram	A-10
A-20.	P_variable Syntax Diagram	A-10
A-21.	C_variable Syntax Diagram	A-10

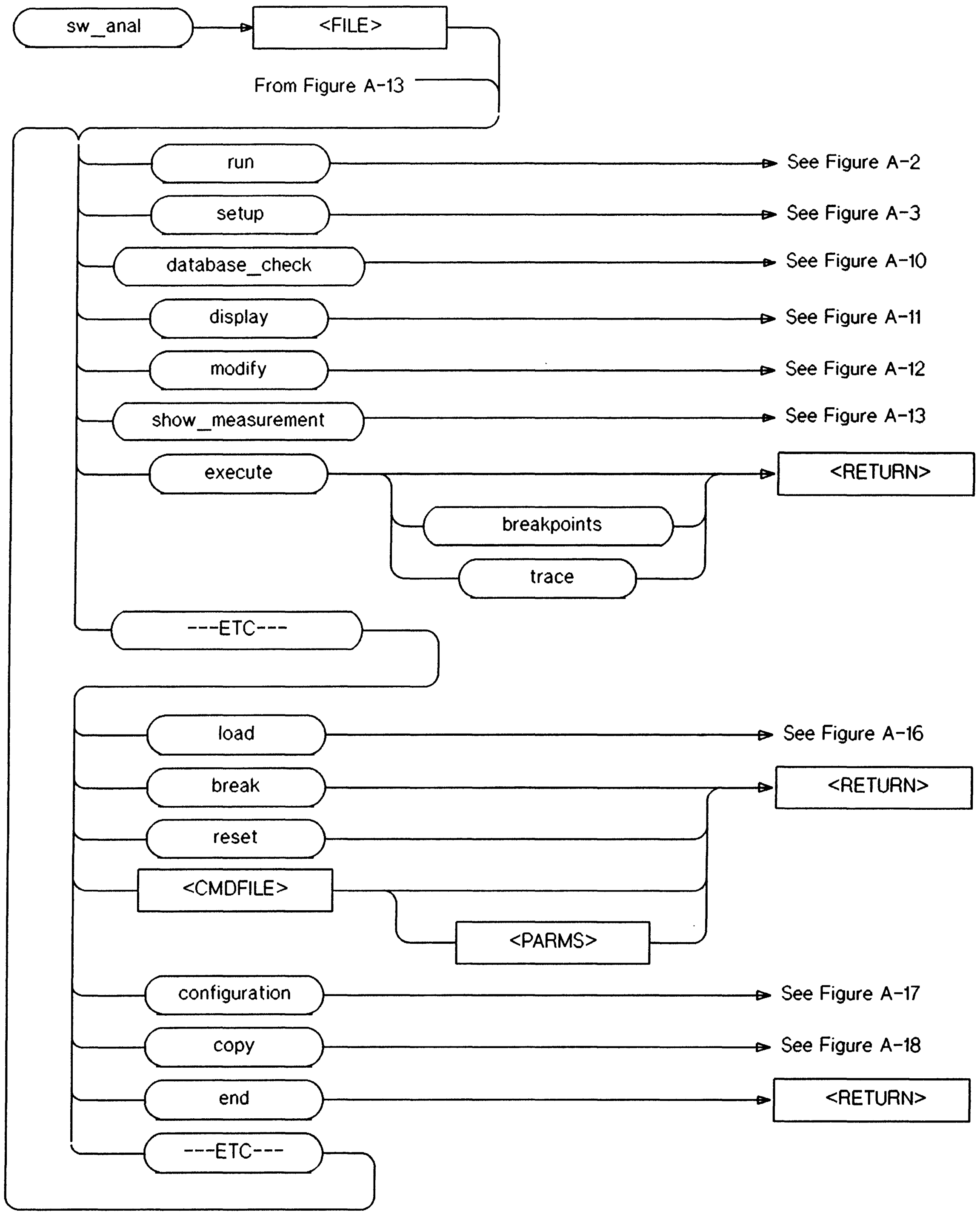


Figure A-1. Software Analyzer Level Syntax Diagram

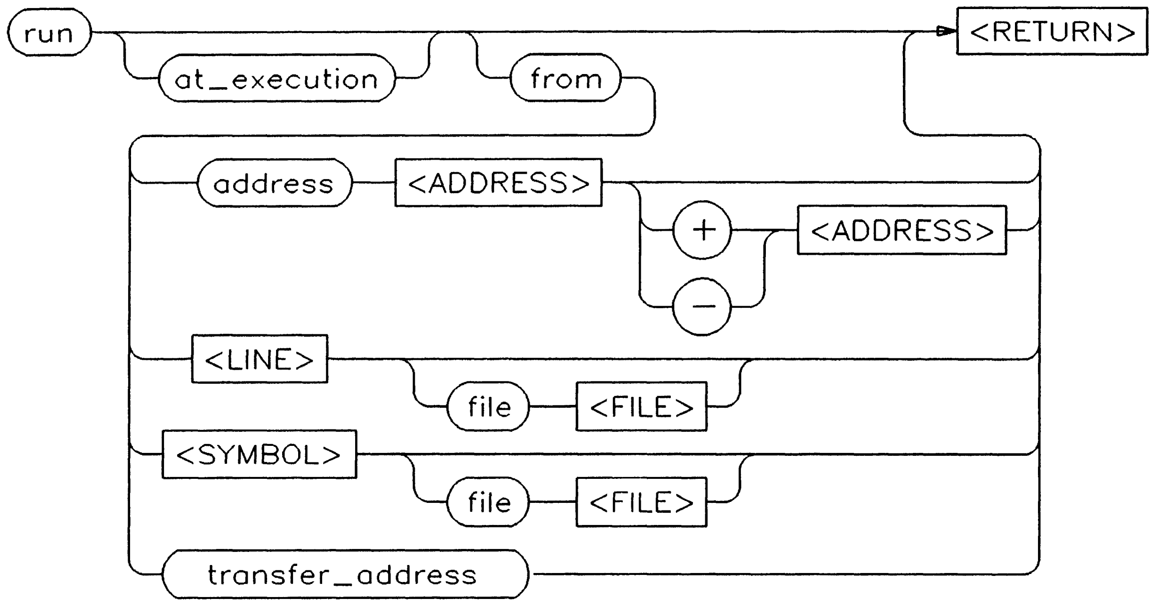


Figure A-2 Run Syntax Diagram

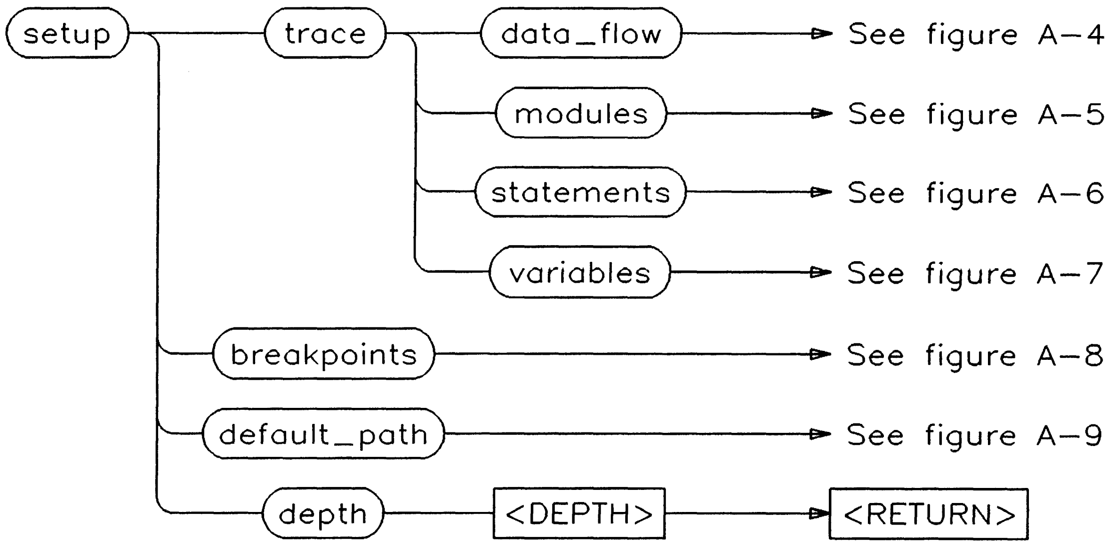
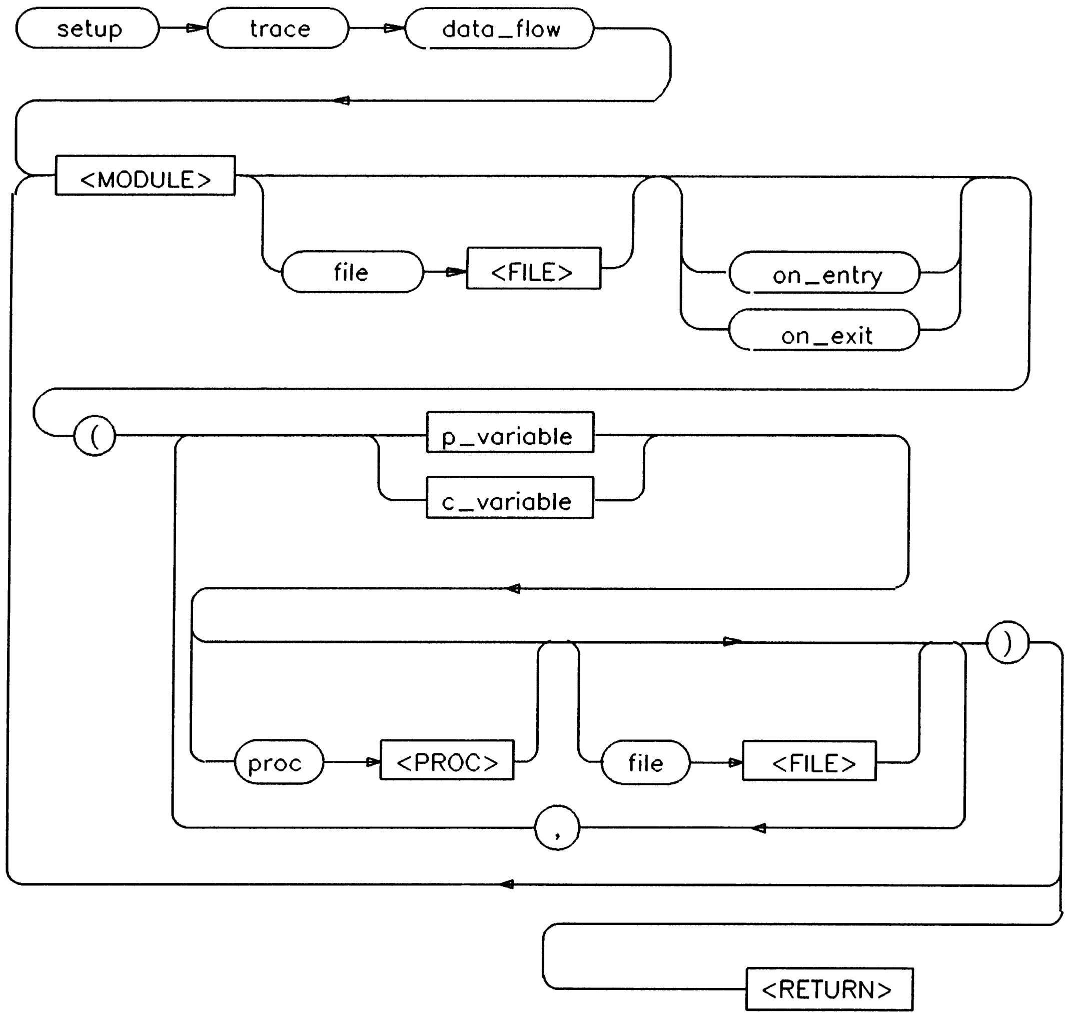


Figure A-3. Setup Syntax Diagram



*See figures A-20 and A-21 for c_variable and p_variable syntax.

Figure A-4. Setup Trace Data_Flow Syntax Diagram

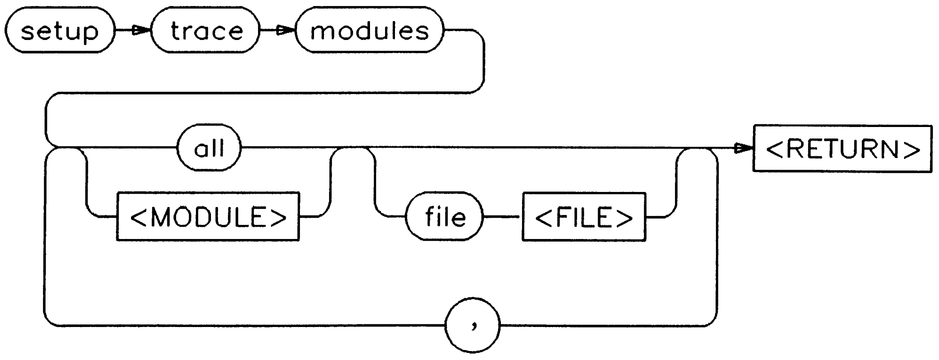


Figure A-5. Setup Trace Modules Syntax Diagram

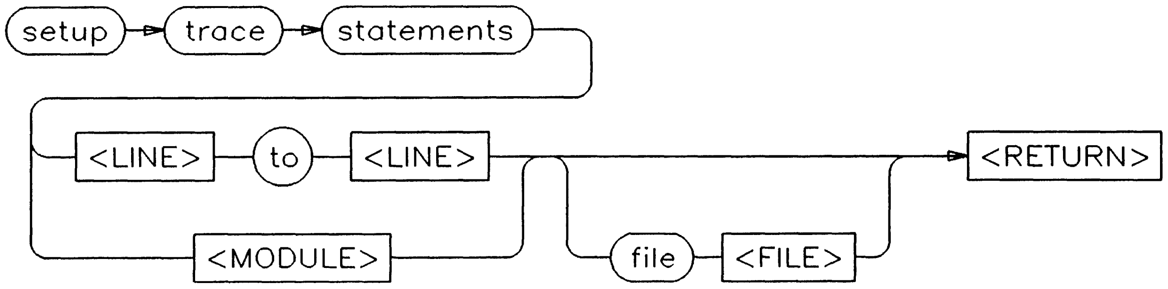
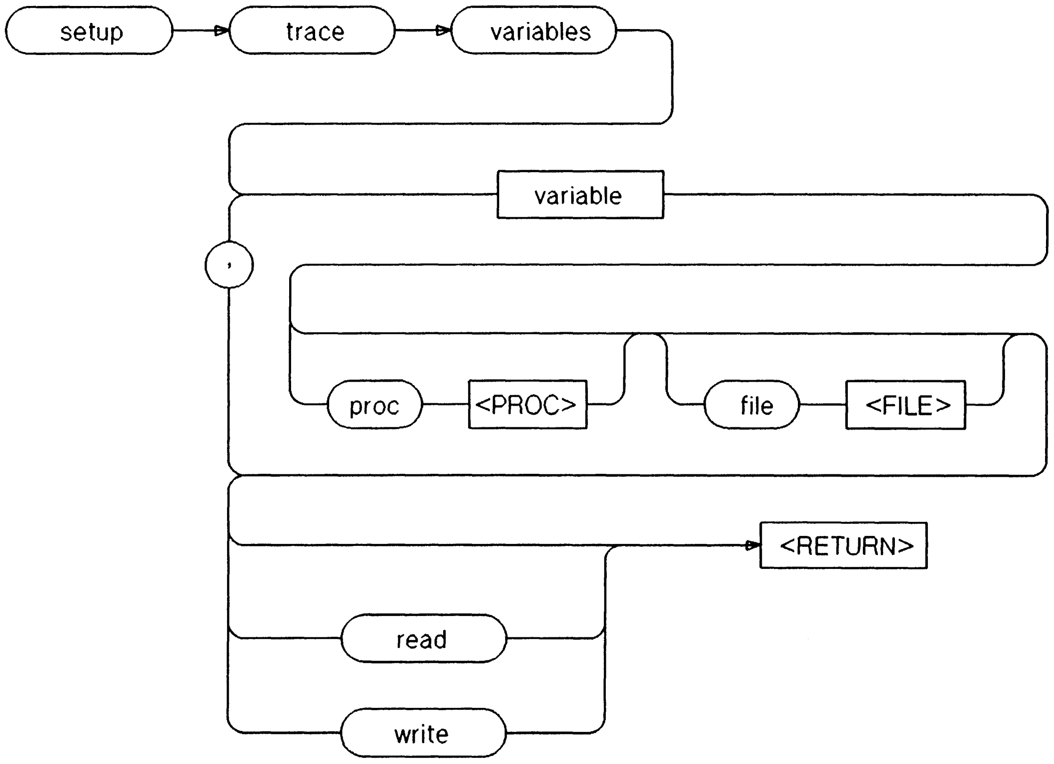


Figure A-6. Setup Trace Statements Syntax Diagram



*See figure A-19 for variable syntax.

Figure A-7. Setup Trace Variables Syntax Diagram

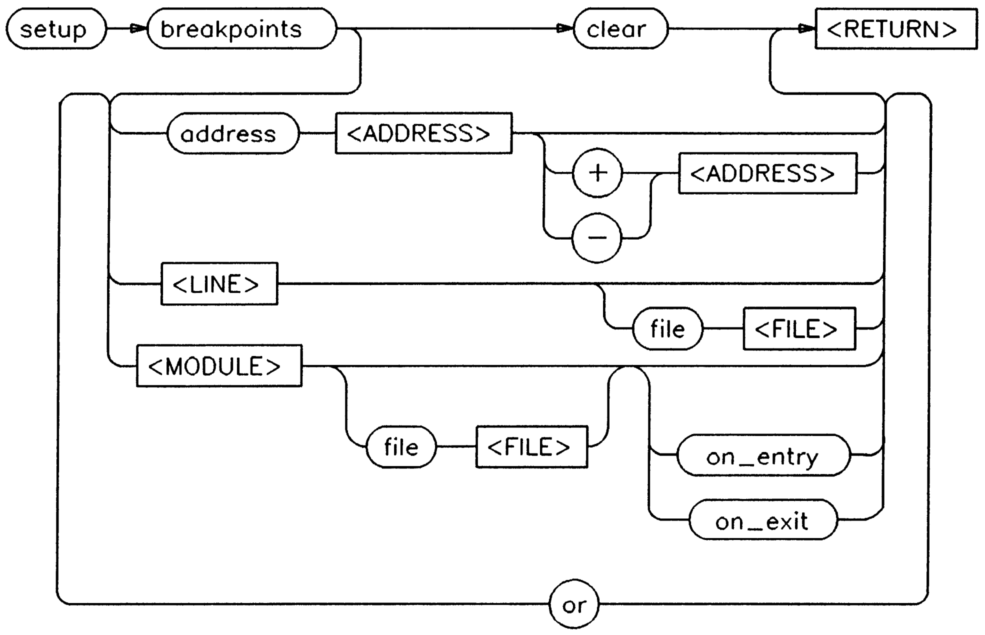


Figure A-8. Setup Breakpoints Syntax Diagram

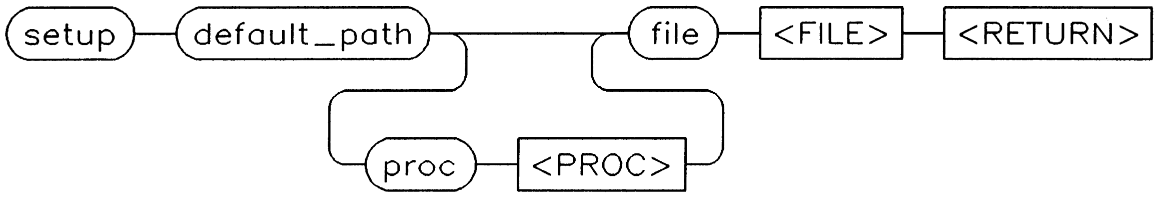


Figure A-9. Setup Default_Path Syntax Diagram

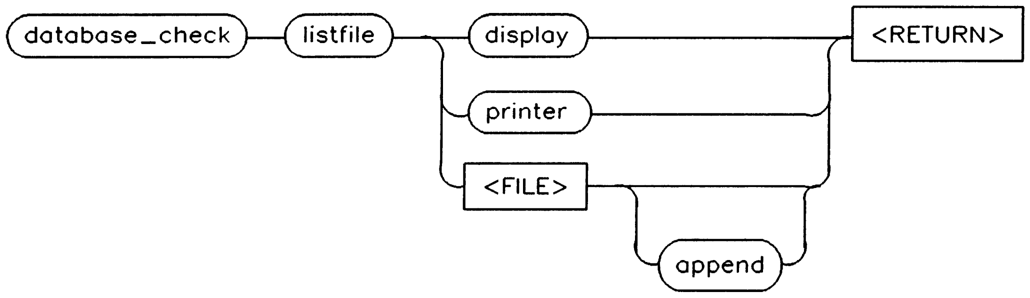
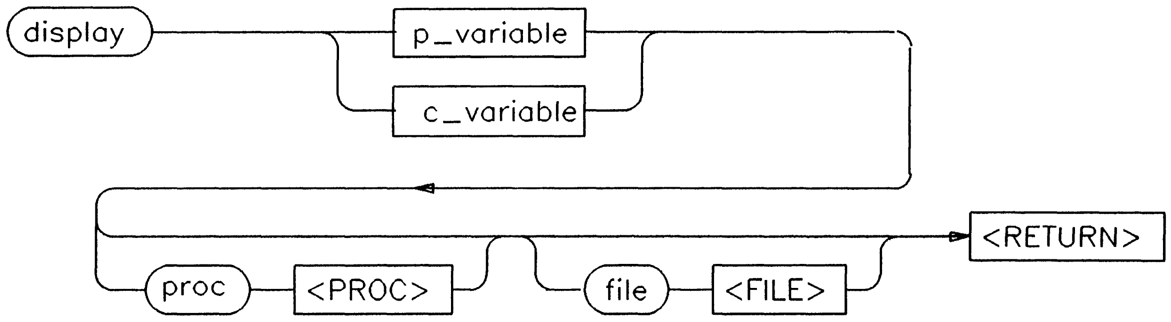
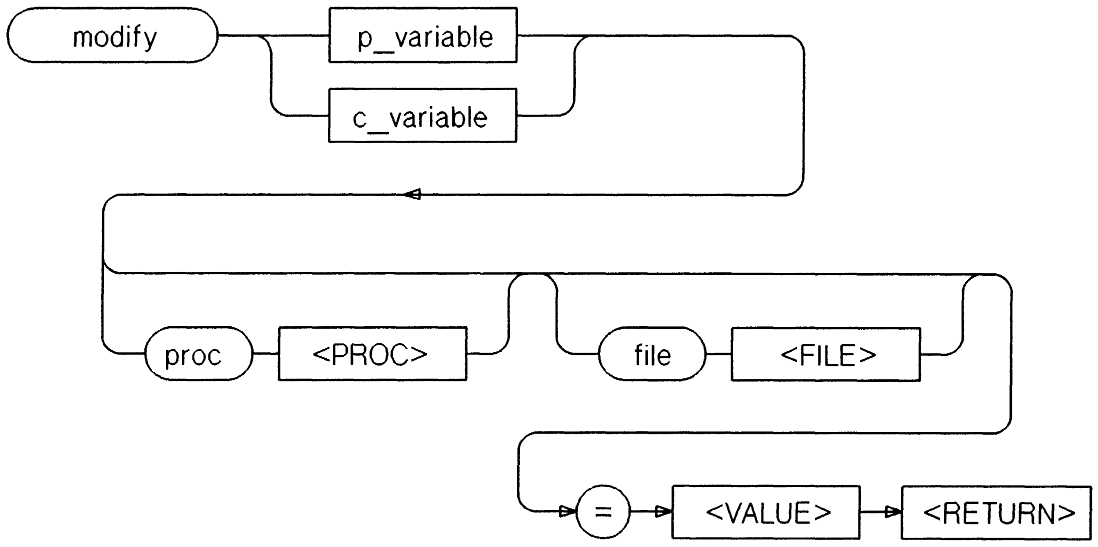


Figure A-10. Database_check Syntax Diagram



*See figures A-20 and A-21 for c_variable and p_variable syntax.

Figure A-11. Display Variables Syntax Diagram



*See figures A-20 and A-21 for c_variable and p_variable syntax.

Figure A-12. Modify Variables Syntax Diagram

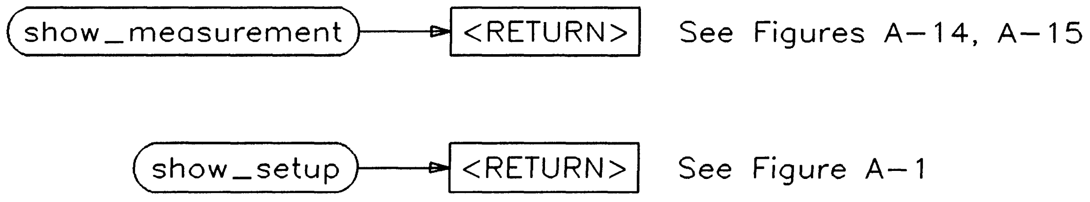


Figure A-13. Show Command Syntax Diagram

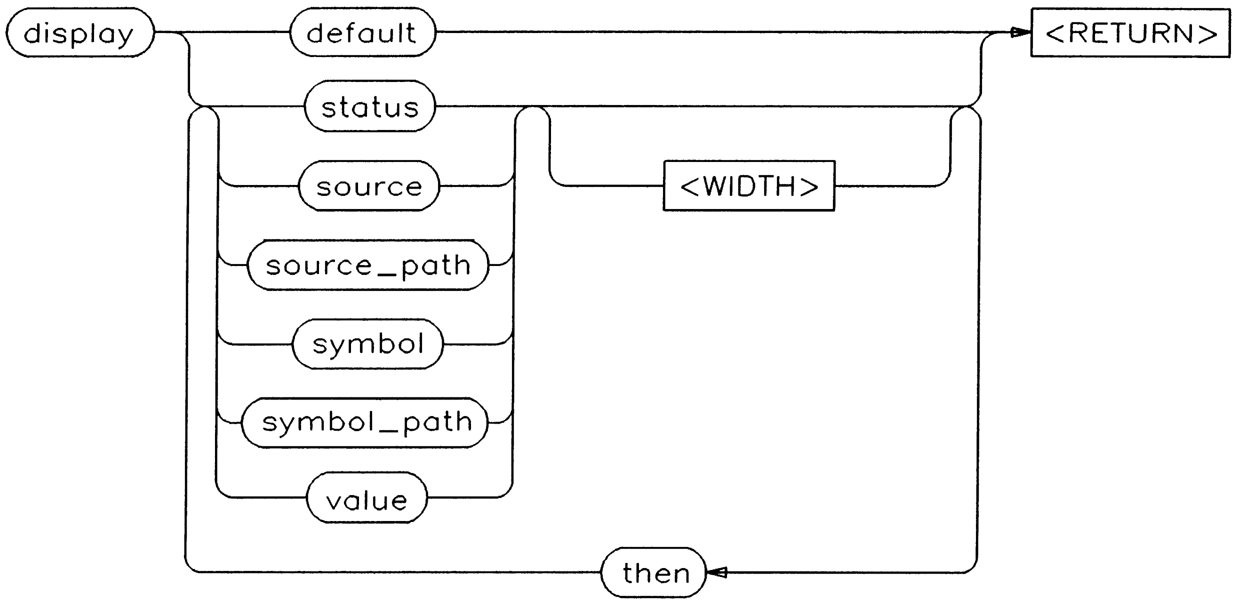


Figure A-14. Display Command Syntax Diagram

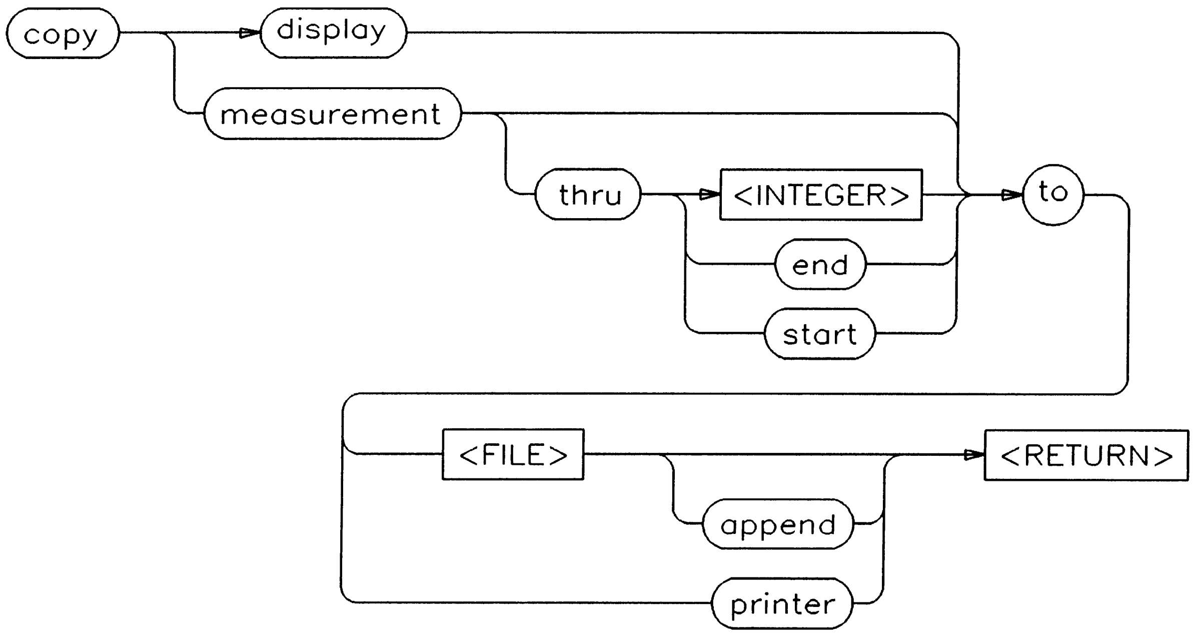


Figure A-15. Copy Command Syntax Diagram (Measurement Display)

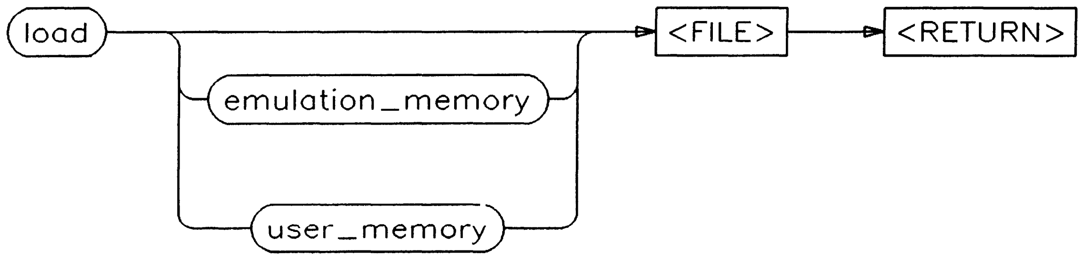


Figure A-16. Load Command Syntax Diagram"

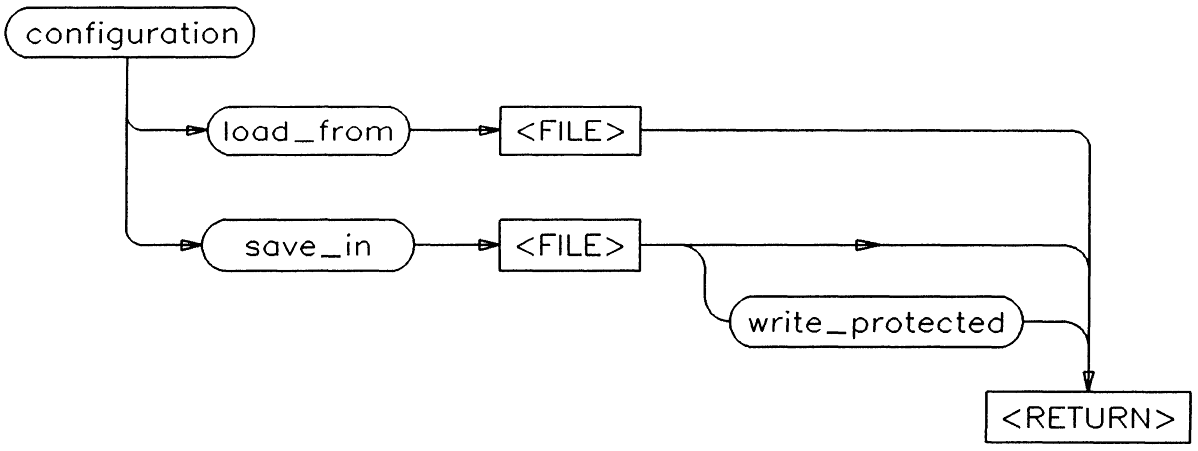


Figure A-17. Configuration Command Syntax Diagram

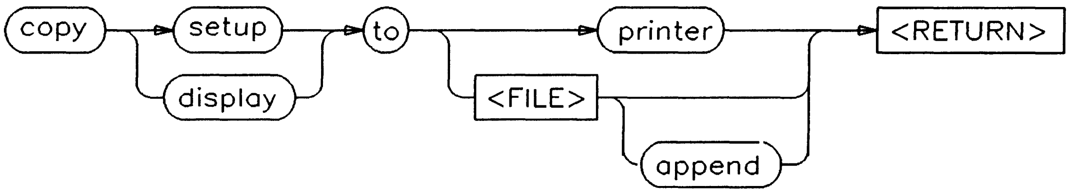


Figure A-18. Copy Command Syntax Diagram (Setup Display)

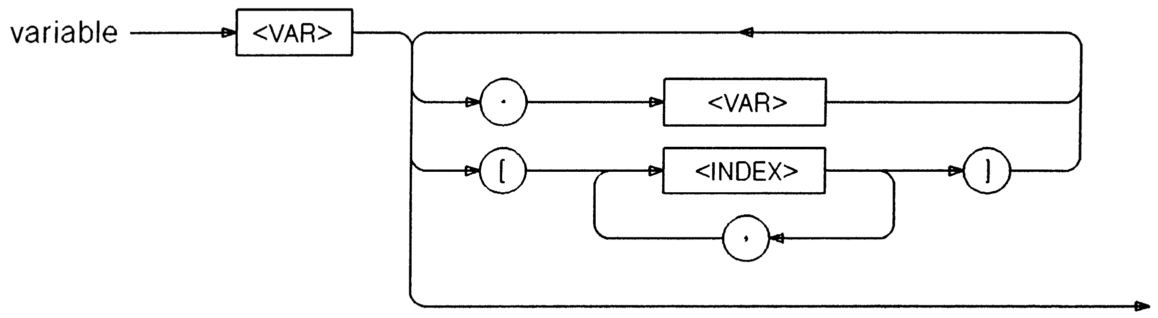


Figure A-19. Variable Syntax Diagram"

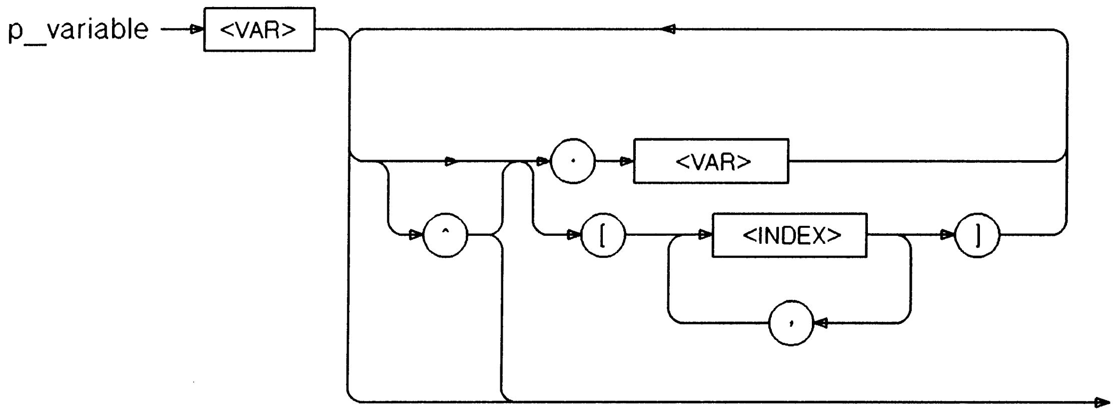


Figure A-20. P_variable Syntax Diagram

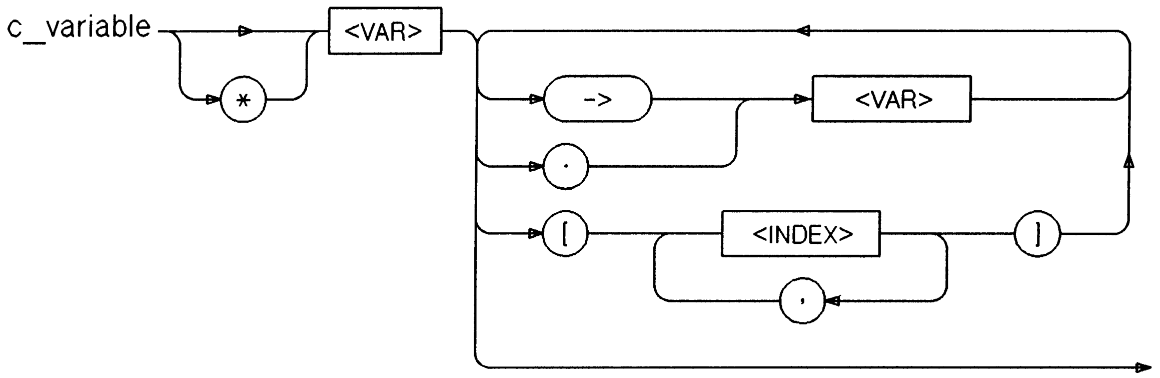


Figure A-21. C_variable Syntax Diagram

Appendix B

STATUS, ERROR AND SOFTKEY PROMPT MESSAGES

INTRODUCTION

This appendix contains a list of the status and error messages, and the softkey prompts and their corresponding messages. All these messages are displayed on the CRT as a result of the software analyzer software. An explanation of each message is given. Table B-1 provides a list of status messages, table B-2 provides a list of error messages, and table B-3 provides a list of the softkey prompts. Status messages are displayed on the screen to provide an indication of operating status. Error messages are displayed on the screen to indicate an improper operating condition or invalid entry on the command line. The softkey prompts are provided on the softkey label line to prompt the user to input the required information.

Table B-1. Status Messages

Status Message	Meaning
Awaiting command	Displayed when the software analyzer is in a quiescent state, ready to accept a new command in its command line. Displayed with measurement display only.
Copying	Displayed when the software analyzer is copying a display, setup, or measurement to a listing file or the system printer.
Copy complete	Displayed when a ***copy*** command has been completed
Database check, files = nn, errors = ee	Displayed during execution of the ***database_check*** command, where nn = the number of files checked and ee = the number of errors found.
Executing, number of states saved = nn	Displayed when a measurement is in progress. Number of states represents the number of data bytes currently acquired in the trace buffer.
Formatting display	Displayed while the acquired data is being formatted for display after completion of a measurement or after execution of a ***display*** command.

Table B-1. Status Messages (Cont'd)

Status Message	Meaning
Formatting next page	Displayed while the next page of acquired data is being formatted for display after the NEXT PAGE key has been pressed.
Formatting previous page	Displayed while the previous page of acquired data is being formatted for display after the PREV PAGE key has been pressed.
Loading configuration	Displayed when the software analyzer is in the process of configuring (either upon entry to the analyzer or during execution of a ***configuration load*** command.
Memory load complete	Displayed after execution of ***load*** command, indicating that the absolute file was successfully loaded into memory.
M68010--Reset	Indicates that the emulator was reset when last checked by the analyzer.
M68010--Running	Indicates that the emulator was executing the user program when last checked by the analyzer.
M68010--Running in monitor	Indicates that the emulator was running in the emulation monitor routine when last checked by the analyzer.
Save complete	Displayed upon completion of a ***configuration save*** command.
Saving configuration	Displayed when the software analyzer is in the process of saving a measurement configuration during execution of a ***configuration save*** command or an ***end*** command.
Trace complete	Displayed upon completion of trace measurement.
Trace stopped	Indicates that the ***halt*** command has been executed to stop the measurement.
Access guarded mem.	Displayed when Memory is accessed that was not mapped in the emulation configuration file.
Access to variable: <VAR> not allowed at procedure	Indicates that the specified variable cannot be accessed at the requested procedure entry or exit point.

Table B-2. Error Messages

Error Message	Meaning
Asmb_sym file contains bad data	Indicates that the asmb_sym file does not contain valid data. Recompile your source file.
Bad file name, reenter file name	Displayed when the file name entered is not in the correct syntax for a file name.
Bad line range	Indicates the line numbers specified are at the same address or the second line number's address is less than the address of the first line number.
Break address is in different file than variable	Indicates that the current PC is not in the same file as the dynamic variable.
Break not allowed from reset	Indicates that break cannot be executed when the emulator is reset. The emulator must be running.
Break while in range, possible missing data	Indicate that the analyzer executed a break and that some variable reads or writes may be missing from the acquired data.
Breakpoints not set up	Displayed when no breakpoints are set up and an ***execute breakpoints*** command has been issued.
Command causes break-runs restricted to real time	Displayed when the command would require the emulator to break, and the emulator command file has configured the emulator for real time execution only.
Comp_db file contains bad data	Displayed when the comp_db file does not conform to the format required by the analyzer. Relink the absolute file using "options comp_db".
Configuration file corrupt	Indicates that the configuration file specified to be loaded is corrupted and cannot be used.
Disc number out of range, reenter file name	Indicates that the disc number specified is not in the range 0 to 7.
Duplicate module names in this file	Indicates that the file contains two modules with names that are identical in the first 15 characters. Duplicate module names are not supported by the software analyzer. The file must be modified to eliminate the duplicate names.
Duplicate module names specified	Displayed when a module is specified multiple times in the ***setup trace*** specification. Modify trace setup.

Table B-2. Error Messages (Cont'd)

Error Message	Meaning
Emul_com file <FILE> has no analysis defined	Displayed when the emulator command file does not define a 64302A Analyzer.
Emul_com file <FILE> is inconsistent with hardware	Displayed when the locations and/or board types in the station do not match the specification defined in the emulation command file.
Exit captured without matching entry information	An exit to a procedure occurred without the analyzer being able to gather information about the stack. In ***trace variables*** mode, any accesses to local variables occurring before the exit were missed. In ***trace data_flow*** and ***trace statements*** modes, any data occurring before the exit was missed.
File exists, wrong module type	Indicates that a file of type "trace" with the specified name exists, but is not a software analyzer configuration file.
File <FILE>:<USERID>:comp_db is outdated	The file <FILE>:<USERID>:comp_db is not supported by the current version of the analyzer. Relink the absolute file containing <FILE>:<USERID> with ***options comp_db*** to update the comp_db file.
File is write protected	Indicates that the configuration file is write protected and cannot be overwritten with another configuration file. To remove the configuration file, you must purge it.
File not found file= <FILE>:comp_db	Displayed when no comp_db file exists for <FILE>. Compile <FILE> using ***options comp_sym***, then relink the absolute file using ***options comp_db***.
File not found file= <FILE>:comp_db (PC= nnnnH)	Displayed when no comp_db file exists for <FILE>. <FILE> may be an assembly language file or a file for which no comp_db file was created. The PC address is the address within the file that caused the access to occur.
Invalid field for this measurement	Displayed when the display field selected is not valid for the current measurement.
Invalid type encountered	The type of the specified variable is not supported by the software analyzer.
Limit for breakpoints is 16	Indicates that the maximum number of software breakpoints allowed is 16.

Table B-2. Error Messages (Cont'd)

Error Message	**Meaning**
Line number <LINE> not found	Indicates that <LINE> has no code associated with it.
Line numbers are not in the same module	Indicates that the two line numbers specified must reside in the same module.
Measurement data not available	Indicates that no measurement data has been acquired. A measurement must first be executed before a *show_measurement* command can be executed.
Measurement in progress, unable to run sw_anly until halted	Displayed when an IMB measurement is in progress. It must be halted or completed before the software analyzer can be used.
Module contains no code	The module specified to be traced contains no user code.
No absolute file loaded	No absolute file is loaded to the emulator. You must load an absolute file before you can execute a measurement.
No measurement setup	Displayed when an *execute* command is given when no measurement setup is defined. A measurement setup must be defined before an *execute* command can be given.
No memory expander board present	Indicates that no memory expander board is installed in the 64000 frame. The 64000 frame has a serial number prefix lower than 2309A and requires a 64030A Memory expander board to run the software analyzer.
No modules in file(s)	Indicates that the file or files requested to be traced contain no modules.
No module name found (PC= nnnnH)	Indicates that the data base, absolute file, or source file has been modified since the measurement was made. PC indicates that a program counter was executed which does not map to a module name.
No monitor program	Displayed when no emulation monitor program was linked with the absolute file or no absolute file was loaded.
No path defined	Displayed when no path has been defined for a symbol. A symbol must have a path definition specified, either in the default path specification or as part of the symbol specification in the command line.

Table B-2. Error Messages (Cont'd)

Error Message	Meaning
No source line found (PC= nnnnH)	Indicates that the PC is part of a source file but has no source line associated with it (e.g. the overhead generated by the compiler for procedure entries when the procedure is the first executable code in the file).
Procedure contains no code	Displayed when the procedure entry address is equal to the procedure exit address.
Processor not in monitor	Indicates that the emulator is not running in the monitor routine.
Processor not supported	Indicates that the software analyzer does not support the processor for which the current emulation command file is set up.
Program execution outside of absolute file (PC= nnnnH)	Displayed when the emulator is executing code outside of the address space defined for the absolute file.
Range traced includes monitor	The range of variables to be traced includes either monitor static variables or locations on the stack that the monitor accesses. The analyzer cannot trace these memory locations, causing the software analyzer to halt the measurement.
Recursive limit exceeded - additional levels not captured	A recursive call was made that exceeded the maximum levels of recursion traced by the software analyzer. Any data occurring from that recursion level until returning to the same level was not traced.
Simulated I/O requested	The target processor has requested an I/O operation. The measurement has been halted. To resume program execution, exit the software analyzer and enter the emulator.
Subelement for variable <VAR> not found	Displayed when the variable expression specified is not found for the variables specified
Symbol is too big, modify subelements individually	Displayed when the requested variable is greater than 4 bytes. modify variable in smaller units.
Symbol: <SYMBOL> is not found	Displayed when the specified symbol cannot be found in the procedure and/or file indicated.
Symbol: <SYMBOL> is not the correct type	Displayed when the symbol specified is not a variable type that can be used in the specified context such as specifying an array index on a pointer variable.

Table B-2. Error Messages (Cont'd)

Error Message	**Meaning**
Symbol(s) type information is too big	Indicates that the type information describing the symbol(s) to be traced or displayed exceeds the maximum size allowed.
Symbols not accessible, absolute file not loaded	Symbols cannot be referenced in a command until the absolute file containing the symbols is loaded. Load the absolute file before referencing symbols in a command.
Syntax error in file name, reenter file name	Indicates that the command entered does not contain the valid syntax for a file name. Reenter the file name.
Syntax invalid	Displayed when an attempt is made to execute an invalid command.
The type for variable <VAR> is not supported	Indicates that the specified variable's type is not supported by the software analyzer.
Too many symbols specified	Indicates that more than the maximum number of symbols are specified in the trace setup.
Unable to access monitor	Displayed when the host processor is unable to communicate with the emulator's monitor program.
Variable access not allowed from current PC	Indicates that the variable is a local variable and is not scoped to this location in the program.
Variable <VAR> has an undefined size	The variable is a C array with unspecified size and cannot be traced.
Variable expression has too many indirections	The variable expression exceeded the maximum number of indirections supported by the software analyzer.
Warning: possible performance penalty, variables not adjacent	Displayed when the variables to be traced are not adjacent to each other in memory. Therefore addresses between the variables must also be checked, causing a possible increase in the time required to acquire data.
Warning: value parameter will not be displayed on module exit	Indicates that a value parameter was requested in a *trace data_flow* measurement on a procedure exit. The variable is not active on procedure exit and cannot be displayed.
Write to ROM	Indicates that a write to a memory location that has been mapped as ROM has occurred.

Table B-3. Softkey Prompt Messages

Softkey Prompt	Message and Meaning
<ADDRESS>	***Any address constant*** <ADDRESS> is any valid address expression representing an address location within the absolute file loaded into user or emulation memory.
<CMDFILE>	***A command file name*** <CMDFILE> prompts the user to enter the name of a command file containing valid software analyzer commands to automatically configure the analyzer or execute a measurement.
<DEPTH>	***Acquisition depth (an integer value 1..8192)*** <DEPTH> is displayed in the ***setup depth*** command and represents the number of bytes allocated for the measurement data acquisition file.
<FILE>	***file_name[:userid][:disc #]*** When used with the ***load*** command, <FILE> is the name of the absolute file to be loaded from the 64000 system memory in user RAM or emulation memory.
<FILE>	***file_name[:userid][:disc #]*** When used with the ***copy*** or ***database_check*** command, <FILE> is the name of the listing file that the display, setup, measurement, or database information is to be copied to.
<FILE>	***A source file name. Note: ':'may replace 'file'*** <FILE> is an optional parameter that refers to the source file containing the specified <MODULE>, <VAR>, <PROC>, or line range called out in the command statement. If the <MODULE>, <VAR>, <PROC>, or line range is in the defined default path, the <FILE> parameter may be omitted from the command statement.
<INDEX>	***An index value or scalar*** An index value (integer or scalar value) specifying a component of an array.

Table B-3. Softkey Prompt Messages (Cont'd)

Softkey Prompt	Message and Meaning
<INVALID>	*Command syntax is invalid* The <INVALID> prompt indicates that the portion of the command between the beginning of the command line and the cursor contains a syntax error. Refer to the softkeys at each point in the command to verify the syntax.
<LINE>	*A program line number* <LINE> represents the line number of a Pascal or C statement in the source program. IF the specified <LINE> contains only comments (no executable code), the analyzer will associate the line number with the first line containing executable code following it. Any comment lines preceding the first line of executable code in a procedure or function are not recognized by the software analyzer. All lines in the specified line range must be contained within a single module. This module may be a procedure or function in Pascal or a function in C, or the main program block.
<MODULE>	*A program or module name* <MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure or a function within a specified file. In C, a module can be the name of a function within a specified file.
<PARMS>	*Command file parameters* The parameters passed to a command file.
<PROC>	*A procedure name. Note: '@' may replace 'proc'* <PROC> is an optional parameter that refers to a procedure or function. If <PROC> is defined in the *setup default_path* command, it may be omitted in the command line. If <PROC> is not specified in either the default path or the command, the analyzer assumes that <VAR> is a global variable defined at the main program level.
<RETURN>	*Command syntax is valid to cursor.* The portion of the command between the beginning of the command and the cursor contains no errors in syntax and can be entered if no further options are desired.

Table B-3. Softkey Prompt Messages (Cont'd)

Softkey Prompt	**Message and Meaning**
<STATE #>	***An integer value*** A positive integer value within the range 0 thru 9999 used to select a position in the measurement data for copy or display. The value specifies the number of bytes offset from the start of the measurement data.
<SYMBOL>	***A valid global or local (specify file) symbol*** <SYMBOL> allows the user to specify program execution to run from a specified symbol. If a file name is specified with <SYMBOL>, the analyzer assumes that the symbol is a module in the specified file. If no file is specified with <SYMBOL>, the analyzer first looks for the address of a global symbol in the link_sym file associated with the currently loaded absolute file. If no global symbol is found there, the analyzer then searches for a module in the current default file.
<VALUE>	***An integer value (must be 32 bits or less)*** <VALUE> represents the value that the specified variable is to be changed to. <VALUE> must be specified as a integer value.
<VAR>	***A valid variable identifier*** Used with the ***setup trace data_flow***, ***setup trace variables***, and ***display variables*** commands, <VAR> represents the name of a valid program variable or parameter. <VAR> can be any valid Pascal or C variable expression.
<VAR>	***A valid variable identifier (must be an intrinsic scalar type)*** Used with the ***modify variable*** command, <VAR> represents the name of a valid program variable of an intrinsic scalar type (single value of 32 bits or less such as one component of an array, not an entire array).
<WIDTH>	***An integer width from 0 to 132 columns*** An integer value from 0 to 132 used with the ***display*** command to specify the width in columns of the specified field.

Appendix C

STACK ARCHITECTURE AND MEMORY STRUCTURE

INTRODUCTION

In order for high level software analysis to be performed on more than one processor it is necessary to tailor the analyzer software to the particular processor. Processor specific capabilities are needed for several reasons. Primarily, since most processors have different architectures and instruction sets, compiler designers will often define the stack architecture for their compiler based on the characteristics of the processor the compiler is targeted for. Consequently, the information required for high level analysis that must be obtained from the stack is not the same from processor to processor. Additionally, the location of compiler generated code used to build the stack varies from compiler to compiler based on the compiler designer's decisions. This affects the way in which the stack reference point is established for analysis, i.e., if the stack is built prior to calling a procedure, upon entry to the procedure the stack pointer is located in a different relative position on the stack than if the stack is built after entry to the procedure.

In addition to the stack architecture, physical memory allocation can vary from one processor to the next making it necessary to be able to accurately interpret the basic structure of the memory. These requirements indicate the need for software personality modules designed to interpret the stack structure of each compiler so that location of parameters, variables and parental information is readily obtainable during analysis of program execution as well as to interpret the physical memory allocation for the processor.

STACK ARCHITECTURE

Pascal Compiler Considerations

The stack architecture for the Pascal compiler is illustrated by the stack frame diagram in figure C-1. A stack frame is created for each procedure call during program execution. Register A6 is used as the base pointer for the stack. Register A7 is the stack pointer. The base pointer is established prior to subroutine calls and is used directly as the reference point for obtaining the needed information from the stack. As the data base is being built during link time, stack addressing information is entered in the data base as an offset from register A6. During run time, when the procedure or function is entered, a software break, set up by the software analyzer, occurs and register A6 is read and saved. In this manner, the stack information required by the measurement may be addressed by adding the offset from the data base to the address contained in register A6.

The static link pointer is created only if the procedure described by the stack frame is nested (level 2 or greater). The static link points to the parent routine's static link. If a parameter is passed by reference, its 4-byte address is placed on the stack in the parameter field. If the parameter is passed by value and is shorter than or equal to 4 bytes, its value is placed in the parameter field. If a value parameter is longer than 4 bytes in size, it is placed in the large value parameter field. If a function returns a value longer than 4 bytes in size, the address of the value is placed in the optional function return value address field. Otherwise the value is returned in a register and the field is not created. The variable 1 through variable N fields contain local variables. The previous frame pointer points to the stack frame of the calling routine. The temporary storage buffer is used by the compiler.

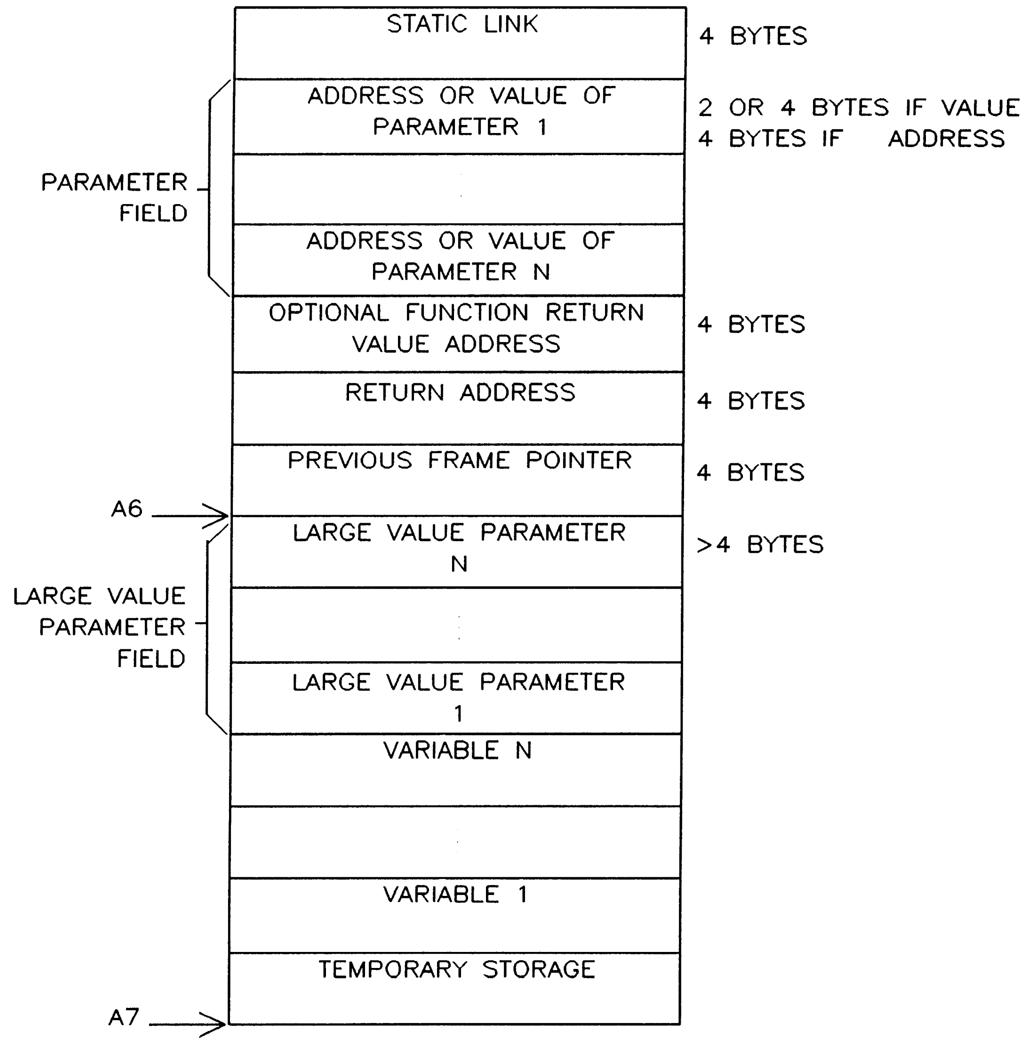


Figure C-1. Pascal Stack Frame

C Compiler Considerations

All parameters in C programs are passed by value, except arrays, which are effectively passed by reference. If arrays are passed as parameters with an unspecified length, they will be unbounded in the compiler symbol table. Consequently, the software analyzer will be unable to trace the parameter "A" if "A" is an array of unspecified length. In order for the software analyzer to trace an unbounded array passed as a parameter, the user must specify a specific element of the array "A[N]" where N specifies an element of an array of unspecified length.

The stack frame structure for the is very similar to that of the Pascal stack frame. With the FIXED_PARAMETERS compiler option ON (figure C-2), the stack frame structure is identical except that, since C does not permit nested functions, no static link field is created. If the FIXED_PARAMETERS option is OFF (figure C-3), the order in which parameters are placed on the stack is reversed.

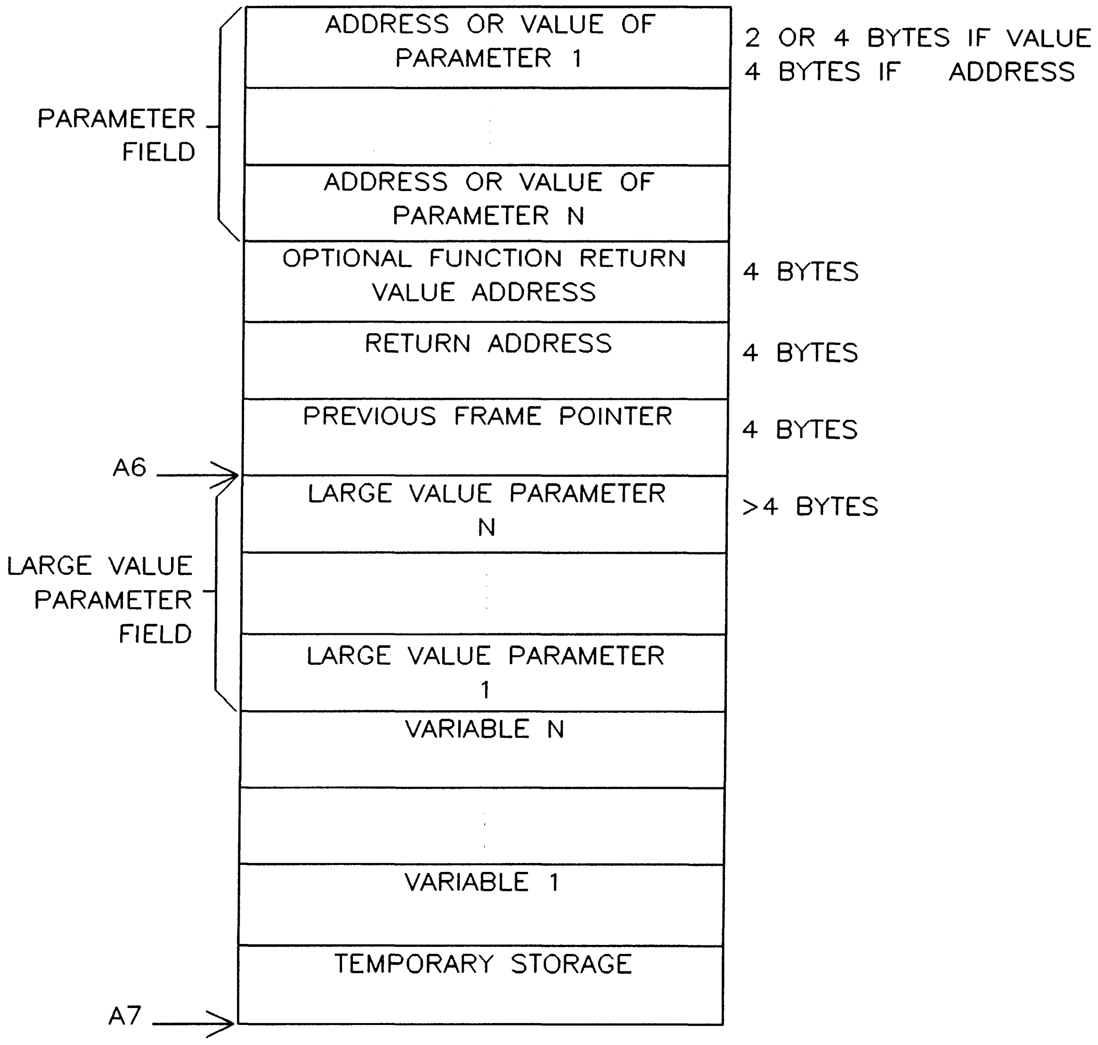


Figure C-2. C Stack Frame (Fixed Parameters Options On)

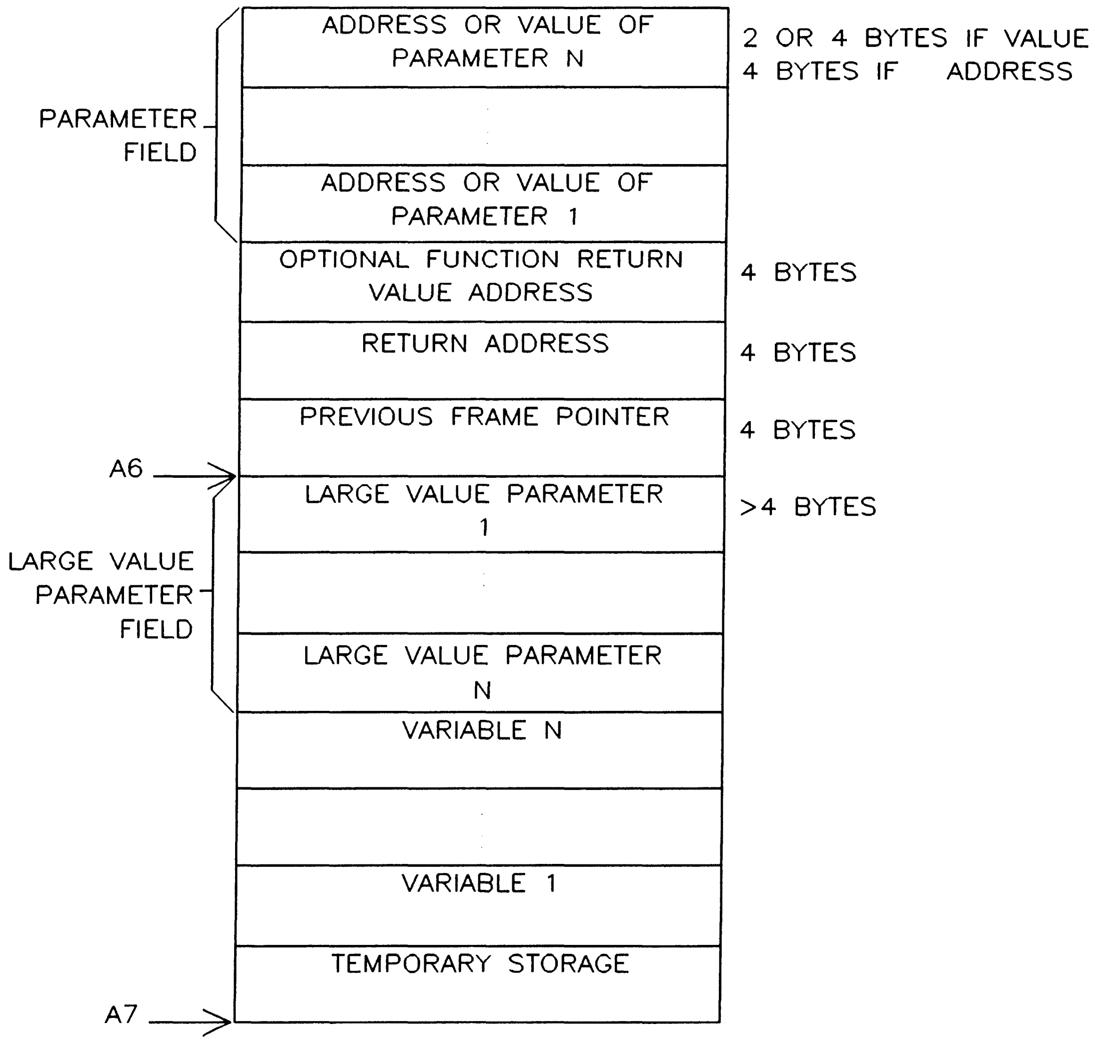


Figure C-3. C Stack Frame (Fixed Parameters Options Off)

Appendix D

GLOSSARY OF SOFTKEY LABELS

INTRODUCTION

Table D-1 contains a list of the softkey labels provided in the software analyzer software. The corresponding command line message is given for each softkey label and an explanation of the softkey label follows. An example is also given which shows the message as it would appear on the command line.

Table D-1. Software Analyzer Softkey Labels

Softkey Label	*Command Line Message*
address	*address*

Used with ***run*** and ***setup breakpoints***, command to indicate that the information that follows is an address constant specified in binary, octal, decimal, or hexadecimal.

```
run from address 2476H
run at_execution from address 3744Q
setup breakpoints address 35FAH
```

all	*all*

Used with the ***setup trace modules*** command to specify that all modules in a file will be traced. If no file is specified in the command, all modules in the default file are traced.

```
setup trace modules all
```

append	*append*

Used with the ***copy*** and ***database_check*** commands to cause information to be appended to an existing listing file.

```
copy display to DSPL append
database_check listfile DBC append
```

at_exec	*at_execution*

Used with the ***run*** command to cause the user's software to begin running in emulation from a specified location at execution of a trace measurement.

```
run at_execution from PROC2
```

Table D-1. Software Analyzer Softkey Labels (Cont'd)

Softkey Label	*Command Line Message*
break	***break***

The ***break*** command causes the emulation break circuitry to force the user's program to be interrupted and the emulation monitor to be entered.

break

breakpts	***breakpoints***

Used in two ways:

1. With the ***setup*** command to define where breakpoints will be inserted in the user program. Up to 16 breakpoints may be defined.

setup breakpoints address 3014H

2. With the ***execute*** command to specify that only software breakpoints are to be executed. If a trace measurement is setup, it will be ignored by the software analyzer.

execute breakpoints

clear	***clear***

Used with the ***setup breakpoints*** command to remove all defined software breakpoints in the setup specification.

setup breakpoints clear

configure	***configuration***

Used to save the analyzer configuration in, or load the analyzer configuration from a file. The file type is trace and contains the entire configuration for the analyzer.

configuration load_from SETUP:USER

copy	***copy***

Used to copy the setup, measurement, or the current display to a listing file or to the printer.

copy measurement to printer
copy setup to FILENAME

Table D-1. Software Analyzer Softkey Labels (Cont'd)

Softkey Label	*Command Line Message*
data_flow	*data_flow*

Used with the ***setup trace*** command to specify the ***trace data_flow*** measurement.

setup trace data_flow PROC1 (X , Y , Z)

db_check	*database_check*

Used to specify a database compatibility check.

database_check listfile display

default	*default*

Used with the ***display*** command to specify that the measurement be displayed in the default format.

display default

depth	*depth*

Used with the ***setup*** command to specify the number of bytes to be allocated for the data acquisition memory. <DEPTH> must be an integer value from 1 to 8192.

setup depth 1024

dflt_path	*default_path*

Used with the ***setup*** command to define a default path (a block or module within a file or a file itself). The default path is used when a path is needed for a measurement and is not included in the measurement command itself.

setup default_path proc PROC1 file FILENAME

display	*display*

Used in four ways:

1. To display the value of variables in a specified procedure and/or file. If no procedure or file is specified in the command, the software looks for the variable(s) in the default path.

display A[3] proc PROC1 file AVER

Table D-1. Software Analyzer Softkey Labels (Cont'd)

Softkey Label	*Command Line Message*
display (Cont'd)	*display*

2. To selectively format and display fields in the measurement display (default, value, source, source_path, status, symbol, and symbol_path).

   display default

3. To request that the display be copied to a file or to the printer.

   copy display to printer

4. To specify the database_check results be listed to the display.

   database_check listfile display

emul_mem	*emulation_memory*

Used with the ***load*** command to specify that the absolute code from the 64000 system disc be transferred into emulation memory. The destination of the absolute code is determined by the address specified during linking.

load emulation_memory AVER

end	*end*

Used in two ways:

1. With the ***copy measurement*** command to specify copying the measurement from the current trace data location (indicated on the status line) through the end of the measurement data.

   copy measurement thru end to printer

2. To end a software analysis session. The current configuration is saved in a file and the software returns to the next higher level, i.e., the measurement system level. The configuration file is named **Sw_anlyX:HP:trace** where X is the cluster address of the station (0 to 7, or to 8 if stand alone).

   end

execute	*execute*

Begins the execution of a measurement.

execute

Table D-1. Software Analyzer Softkey Labels (Cont'd)

Softkey Label	*Command Line Message*
file	*file*

Used to indicate that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the *file* softkey.

setup default_path file TESTP
display SAM proc PROC4 file TESTP

from	*from*

Used with the ***run*** command to specify the location in the user's program from which program execution will begin in emulation.

run from START

halt	*halt*

Used to halt the measurement currently in process. The data collected before the ***halt*** command was executed is displayed.

halt

listfile	*listfile*

Used with the ***database_check*** command to select one of three output devices for listing results; the display, the printer, or a file.

database_check listfile RESULTS append

load	*load*

Used to transfer absolute files from the 64000 system disc into user RAM or emulation memory.

load emulation_memory TESTP

load_from	*load_from*

Used with the ***configuration*** command to configure the software analyzer as specified in the configuration file being loaded. The configuration file is of type trace.

configuration load_from SETUP:USER

Table D-1. Software Analyzer Softkey Labels (Cont'd)

Softkey Label — *Command Line Message*

meas_comp — *measurement_complete*

Used with the ***wait*** command during execution to suspend execution of a command file until the current measurement is completed.

wait measurement_complete

modify — *modify*

Used to modify the current value of variables in emulation or user memory.

modify Q.CHAR1 proc LTRSORT = 41H

modules — *modules*

Used with the ***setup trace*** command to specify that modules (functions or procedures) are to be traced.

setup trace modules all

on_entry — *on_entry*

Used in two ways:

1. With the ***setup trace data_flow*** command to specify that variables be traced only on entry to the specified module(s). The default condition is to trace variables on both entry to and exit from a module.

setup trace data_flow PROC1 on_entry (A[1])

2. With the ***setup breakpoints*** command to specify that the breakpoint be set on entry to a module. The default condition is to set breakpoints on both entry and exit to a module.

setup breakpoints FACTORIAL on_entry

on_exit — *on_exit*

Used in two ways:

1. With the ***setup trace data_flow*** command to specify that variables be traced only on exit from the specified module(s). The default condition is to trace variables on both entry to and exit from a module.

setup trace data_flow PROC2 on_exit (SVN)

Table D-1. Software Analyzer Softkey Labels (Cont'd)

Softkey Label	*Command Line Message*
on_exit (Cont'd)	*on_exit*

2. With the ***setup breakpoints*** command to specify that software breakpoints be setup only on exit from the specified module(s). The default condition is to setup breakpoints on both entry to and exit from a module.

setup breakpoints PROC2 on_exit

or	*or*

Used as a logical combinatoric for inclusive ORing software breakpoints together. Up to a maximum of 16 breakpoints may be specified.

setup breakpoints address 3012H or 122 or MODULE2

printer	*printer*

Used in two ways:

1. With the ***copy*** command to specify that the display, setup, or measurement be copied to the system printer.

copy display to printer

2. With the ***database_check*** command to specify that the results be copied to the system printer.

database_check listfile printer

proc	*proc*

Used to indicate that a procedure or function name follows that defines the procedure or function to which a variable belongs. NOTE: an "@" may be used in place of ***proc***.

setup trace variables FLAG1 proc FUNCTIONX

protected	*write_protected*

Used with the ***configuration save_in <FILE>*** command to prevent the accidental modification of the file with a later ***configuration save_in*** command. The file is protected against writes only within the software analyzer. It can still be purged, renamed, or copied into from the system monitor level.

configuration save_in SETUP:USER write_protected

Table D-1. Software Analyzer Softkey Labels (Cont'd)

Softkey Label	*Command Line Message*
read	***read***

Used to specify that only memory read accesses to a variable be traced. The default condition is to trace both memory read and memory write operations on a specified variable.

setup trace variables QFLAG read

reset	***reset***

Used to suspend emulation system operation and reestablish initial operating parameters. The reset signal is latched when active and is released by the ***run*** command.

reset

run	***run***

If the processor is in a reset state, ***run*** will cause the reset to be released and, if a ***from*** address is specified, the processor will begin program execution at that address. If the processor is running in the emulation monitor, the ***run*** command causes the processor to exit into the user program.

run from address 312EH

save_in	***save_in***

Used with the ***configuration*** command to save the software analyzer configuration in a file. This file is of type trace.

configuration save_in SETUP:USER

setup	***setup***

The ***setup*** command is used to specify the trace measurement, software breakpoints, default path, or the acquisition depth.

setup trace modules PROCEDURE1
setup depth 256

Table D-1. Software Analyzer Softkey Labels (Cont'd)

Softkey Label	*Command Line Message*
show	*show_setup* *show_measurement*

The *show* command is used to transport the user between the setup and measurement displays. *show_measurement* causes the setup display to be replaced by the measurement display on the CRT. *show_setup* causes the measurement display to be replaced by the setup display on the CRT.

show_measurement
show_setup

Softkey Label	*Command Line Message*
source	*source*

Used with the *display* command to specify that the source field be displayed in the trace listing on the measurement display.

display source

Softkey Label	*Command Line Message*
src_path	*source_path*

Used with the *display* command to specify that a source_path field be displayed in the trace listing showing the source file name that the source statement was extracted from.

display source_path

Softkey Label	*Command Line Message*
statements	*statements*

Used with the *setup trace* command to specify that the software analyzer operate in the trace statements measurement mode.

setup trace statements PROCEDURE4

Softkey Label	*Command Line Message*
start	*start*

Used with the *copy measurement* command to specify that measurement results from the start of the trace to the current line be copied to a file or the system printer.

copy measurement thru start to LIST

Table D-1. Software Analyzer Softkey Labels (Cont'd)

Softkey Label	*Command Line Message*
status	***status***

Used with the ***display*** command to specify that the status field be displayed in the trace listing on the measurement display.

display value then status

sw_anly	***sw_anly***

Used in the measurement system level of softkeys to enter the software analyzer. May be followed by an optional software analyzer configuration file name specifying a configuration file from which the analyzer is to be configured from or an emulation command file name of file type emul_com.

sw_anly TRACE_VAR

symbol	***symbol***

Used with the ***display*** command to specify that the symbol field be displayed in the trace listing on the measurement display.

display symbol then status then source

symb_path	***symbol_path***

Used with the ***display*** command to specify that a symbol_path field be displayed in the trace list showing the path in which the symbol is defined. For modules, the symbol_path contains a file name. For variables and parameters, the symbol_path may be a file or a module and file, depending upon the level at which the symbol is defined.

display symbol then symbol_path

then	***then***

Used as a delimiter to separate sequential field specifications in the ***display*** command.

display source then symbol then status

Table D-1. Software Analyzer Softkey Labels (Cont'd)

Softkey Label	*Command Line Message*
thru	***thru***

Used with the ***copy measurement*** command to specify a range of data in the trace listing to be copied to a file or to the system printer. The minimum amount of data copied is the contents of the current display.

copy measurement thru end to printer

to	***to***

Used in two ways:

1. With the ***setup trace statements*** command to specify a line range to be traced in a source program. All lines in the specified range must be contained in a single module.

setup trace statements 57 to 86

2. With the ***copy*** command to specify either a listing file or the system printer.

copy display to printer

trace	***trace***

Used in two ways:

1. With the ***setup*** command to specify the trace measurement mode to be executed by the software analyzer. The measurement modes are ***trace data_flow***, ***trace modules***, ***trace statements***, and ***trace variables***.

setup trace statements PROCEDURE1 file TESTP

2. With the ***execute*** command to specify that only the trace measurement is to be executed. The software breakpoints setup will not be executed.

execute trace

transfer	***transfer_address***

Used with the ***run*** command to specify that the emulator begin program execution at the address stored in the transfer buffer (XFR_BUF). This is the starting address of the user program.

run from transfer_address

Table D-1. Software Analyzer Softkey Labels (Cont'd)

Softkey Label	*Command Line Message*
user_mem	*user_memory*

Used with the ***load*** command to specify that the absolute program be loaded into user RAM in the target system.

load user_memory FILENAME:USER

value	*value*

Used with the ***display*** command to select the value field for display in the trace measurement listing.

display value

variables	*variables*

Used with the ***setup trace*** command to specify that the software analyzer operate in the trace variables mode.

setup trace variables COUNT proc PROC2

wait	*wait*

Causes execution of a command file to be suspended until the current measurement being executed is completed. ***measurement_complete*** option must be used with the ***wait*** command.

wait measurement_complete

write	*write*

Used with the "setup trace variables" command to specify that only memory write accesses to the variable be traced.

setup trace variables SNN proc FUNCTION write

Appendix E

RESOLVING MEASUREMENT PROBLEMS

INTRODUCTION

This appendix describes measurement problems you may encounter while using the software analyzer, their possible causes, and suggested solutions. Measurement abnormalities may result from the use of certain compiler directives, improper use of compiler and linker options, use of breakpoints, how measurements are implemented in the software analyzer, and many other causes. This appendix lists the most common problems, their causes, and suggested solutions to the problem. Programming style can also affect how the analyzer traces data. The section, ***Recommended Programming Style***, in chapter 3 gives guidelines for writing code to achieve the best results from your software analyzer.

MEASUREMENT PROBLEMS AND SOLUTIONS

Missing Source Statements

Problem: An expected "go to" statement is not displayed in trace statements. The analyzer may miss a "go to" statement if a multiword access is on the preceding source line.

Solution: Execution of the "go to" is seen in the change of source line numbers. To see the "go to" statement, structure the code so that a multiword access is not on the preceding source line.

Problem: A statement containing an "end" (in Pascal) or "}" (in C) is not displayed. The analyzer uses the "end" or "}" to indicate the end of user code. This can cause unexpected results in the trace display.

Solution: Place the "end" statement of a procedure in Pascal or the "}" terminator in a C function on a separate line containing no code.

Problem: Variable accesses in a statement containing an "end" (in Pascal) or "}" (in C) are not displayed. The analyzer uses the "end" or "}" to indicate the end of user code. This can cause unexpected results in the trace display.

Solution: Place the "end" statement of a procedure in Pascal or the "}" terminator in a C function on a separate line containing no code.

Missing Source Statements (Cont'd)

Problem: Missing data during recursive calls in trace modules. Data is not displayed during recursive calls in trace modules when the recursive level is so deep that indentation causes the module not to appear on the display.

Solution: Expand the screen width of the symbol field with the "display" command. In some cases, the number of recursive levels may cause enough indentation that the maximum field width may be exceeded and the data cannot be shown.

Problem: The source line is missing for the entry of the first procedure in a file. The compiler overhead for entry into the procedure is executing. The source line is undefined for these instructions.

Solution: None.

Missing Symbols On The Display

Problem: The software analyzer cannot display a variable maintained in a register.

Solution: Turn the compiler option AMNESIA on.

Problem: The software analyzer cannot display an array parameter in C without an explicitly defined size.

Solution: Declare the maximum size required for the array.

Problem: The software analyzer cannot display the object of a pointer in the trace statements and trace variables measurements.

Solution: Insert a software break in locations where you want to look at the pointer and use the display command. The trace data flow measurement can also be used trace the object of a pointer.

Problem: The software analyzer only displays the first specified variable of differently named variables mapped to the same location.

Solution: When setting up a measurement, specify first the more important variables that you want to see.

Problem: If a breakpoint is encountered in a recursive procedure, data is missed at the beginning of the next trace variables measurement as the program exits from the recursive calls.

Solution: The entry point of the procedure must be seen by the analyzer. Run the program from the point just before the recursive calls begin.

Missing Symbols On The Display (Cont'd)

Problem: Reads or writes to a variable may be missing after the message "***Break while in range, possible missing data***".

Solution: Run the trace again, starting it at a different location.

Problem: Function return values are not displayed.

Solution: The assignments to variables that receive function return values can be traced with the trace statements or trace variables measurement. Using a temporary variable within the function to compute the function value may be a useful way to trace function return values.

Problem: In a trace statements measurement, accesses to variables by the last instructions executed within the address range do not appear with prefetched processors.

Solution: If you are tracing a line range, add a couple of lines to the end of the range. If you are tracing a module, adding a dummy assignment statement at the end of the procedure will solve the problem.

Unexpected Analyzer Execution

Problem: The analyzer doesn't capture any data. The measurement may be set up incorrectly.

Solution: Check ***run*** parameters. Try resetting the emulator and reloading the file.

Problem: The comp_db file is not current with the absolute file.

Solution: Run the db_check and correct the indicated errors.

Problem: The analyzer never finds a breakpoint after a run command has been specified. The breakpoints are only in effect on an ***execute*** command.

Solution: Specify ***run at execution***, followed by an ***execute*** command.

Problem: The number of states acquired is greater than specified in the ***trace statements*** measurement. The software analyzer hardware captures 512 states at a time. It cannot stop acquiring data at exactly the user specified depth.

Solution: None.

Unexpected Analyzer Execution (Cont'd)

Problem: A local variable cannot be modified. The variable is not scoped to the current program counter.

Solution: Insert a breakpoint in the code where it is valid to access the variable, run until the breakpoint, then modify the variable.

Problem: Real, character, or scalar variables can not be modified in their native type. Only variables that are 32 bits or less can be modified and only numerically.

Solution: None.

Unexpected Emulation Operation

Problem: A specify trace or specify run in the emulator does not work correctly after the software analyzer has been used. The software analyzer and the emulator share the same resources (the emulator hardware and the internal analysis card).

Solution: Don't use the specify functions of the emulator concurrently with the software analyzer.

Problem: An error occurs on execution of a measurement after an absolute file containing the emulation monitor program is loaded and the emulator is currently running in the monitor. A load cannot be done correctly while code is running.

Solution: The processor must be reset before loading an absolute file containing the emulation monitor program.

Unexpected Error Or Status Message.

Problem: The program runs until completion in the emulator and in the software analyzer the message "*running*" is the only one given. The software analyzer only checks the status of the emulator during an ***execute*** command, a ***show measurement*** command, when first entering the software analyzer, or on a ***configuration***, ***reset***, ***break***, or ***run*** command.

Solution: Perform one of the above operations to update the status of the emulator.

Problem: The program being traced crashes, the software analyzer gives only the error message "***access to guarded memory***", and no trace data is displayed. Something is fundamentally wrong with the program. No source code is able to execute. For example, the stack may have been placed in a section of memory that was guarded.

Solution: Exit the software analyzer and use the emulation subsystem to discover the problem. Assembly language tracing is required.

Unexpected Error Or Status Message (Cont'd)

Problem: An absolute file was loaded without the emulator being reset.

Solution: Reset the emulator, then reload the absolute file.

Problem: The error message "*access to guarded memory*" appears on execute. The comp_db file is not current with the absolute file.

Solution: Run the db_check and correct the indicated errors.

Problem: The error message "*File not found file= <FILE>:comp_db (PC=nnnnH)*" is displayed and the link process generated no errors or a C program is run and the message appears for FILE "entry". A comp_db file does not exist for the assembly language file. No source data can be returned.

Solution: None.

Problem: An apparently correct variable or procedure name is included in a trace specification and the error message "*: xxx is not found*" is displayed. User error.

Solution: Check all elements of the path given for the variable or procedure. Make sure all elements of the path are there and are correct. Verify that the default path and loaded absolute refer to the same file.

Problem: A "*Bad line range*" error message is displayed at execution of a trace measurement. The lines specified may be at the same address in the program.

Solution: Increment the last line in the range.

Problem: An apparently correct command does not work when a lower case identifier is used. The lower case identifier is identical to one of the software analyzer commands. Identifiers cannot have the same name as software analyzer commands.

Solution: Rename the lower case identifier in the source code.

Problem: The software analyzer does not accept a
configuration file and the error message "*Symbols not accessible, absolute file not loaded*" is displayed. A *run* specification has been setup using a global symbol and saved in a configuration file. When loading loading the configuration file, the address of the symbol cannot be found because the absolute file containing the specified symbol has not been loaded.

Solution: Load configuration files with *run* commands using global symbols only when the absolute file containing the global symbol has been loaded.

Unexpected Error Or Status Message (Cont'd)

Problem: The error message "***Program execution outside of absolute file (PC=nnnnH)***" is displayed and no symbolic information is shown. The software analyzer does not display symbolic information when more than one absolute file is loaded and files in any absolute file other than the last one loaded are being traced. The software analyzer can provide symbolic information only for files in the last absolute file loaded.

Solution: Ensure that the files you want to trace are in the absolute file listed on the software analyzer setup display.

Problem: The error message "***Symbol not found***" is displayed for a known C variable. The specified variable is a block variable (a variable defined within an inter block of a procedure). The software analyzer does not support block variables.

Solution: Declare the variable at the procedure level.

Unexpected Source Line

Problem: The source line is not the line associated with the symbol in a trace variables measurement. The software analyzer shows a prefetched line as causing the access.

Solution: The correct source line is the previous source line displayed in the trace.

Problem: Comments are displayed as source lines. The comment spans multiple lines.

Solution: Start and end each comment line with a comment delimiter.

Problem: The source statement shown at the beginning ot a procedure is an "end" statement from the previous procedure. The compiler overhead for entry into the procedure is executing. The source line is undefined for these instructions, however, the line preceding the procedure declaration is shown.

Solution: None

Unexpected Symbols On The Display

Problem: A variable name is shown on the display that is different than the expected name. When a variable is passed by reference to a procedure, the procedure accesses that variable under an alias.

Solution: None.

Problem: Extra accesses to a record are shown without a field name or with an extra index to an array field. This is caused by pad bytes the compiler inserts in records to align the fields.

Solution: Create records that don't have pad bytes.

Problem: Unexpected reads/writes at the beginning or end of a procedure caused by compiler overhead.

Solution: Check an assembly listing file of the program. These reads/writes may be operations done by the compiler to set up stacks, transfer parameters, etc.

Unexpected Termination of Measurement

Problem: A trace has been started and the "***wait measurement_complete***" command has been issued, but the measurement is not completing. There is no way to halt a trace measurement after the "wait" command has been given.

Solution: Execute the (RESET) command twice to return to the monitor.

Unexpected Value On The Display

Problem: The value of the symbol is illegal for the symbol type.

Solution: None.

Problem: Incomplete access to variables. This can occur when the software analyzer is tracing two 32-bit integers and one is assigned to the other or when the software analyzer is tracing two structured variables where one is assigned to the other.

Solution: The complete value can be determined by noting the partial values and their locations. Alternatively, set up a breakpoint on the next source line after the assignment and display the variable.

Unexpected Value On The Display (Cont'd)

Problem:	Partial values are displayed or incorrect complete values in the trace variables measurement. The trace data begins part of the way through a variable access or the user positions the display to the middle of a variable access.
Solution:	Use the display positioning to find a location that makes the partial values disappear. If that doesn't work, piece together the whole value from the partial values. Alternatively, setup a breakpoint on the next source line after the assignment and display the variable.

Appendix F

SPECIFICATIONS

INTRODUCTION

This appendix lists the specifications for the software analyzer. The specifications are given in table F-1.

Table F-1. Software Analyzer Specifications

Trace Measurements:	
Trace Variables:	10 symbols maximum (including both variable and module names)
Trace Data_flow:	10 symbols maximum (including both variable and module names)
Trace Statements:	1 module or line range maximum
Trace Modules:	128 modules maximum
Recursion Levels:	128 levels maximum traced for all measurements except trace modules (unlimited levels for trace modules).
Software Breakpoints:	16 maximum (breakpoint on module entry and exit requires two breakpoints)
Modify Variables:	1 32-bit value maximum per command
Display Variables:	1 maximum per command
Trace Depth:	8192 maximum
Language Support:	Pascal, C

NOTES

INDEX

a

absolute file (PC=nnnnH) displayed E-6
absolute file loaded without emulator being reset E-5
accesses to variables not displayed E-3
accessing the software analyzer 3-8
acquisition depth 3-13
acquisition memory depth 6-1
active variable 6-3
address D-1
<ADDRESS> B-8
all 6-8,D-1
analyzer does not accept configuration file E-5
analyzer does not capture data E-3
analyzer does not find breakpoint E-3
apparently correct command does not work E-5
append 8-8,D-1
array parameter in C not displayed E-2
arrays 11-5
ASMB_SYM 4-7
assembly language 4-8
at_execution D-1

b

base type 11-5
boolean 11-3
break 1-7,11-1,D-1
break command 5-4
break softkey 3-2
breakpoints 1-7,7-1
breakpts D-2
building database files 3-3,4-1
building the symbol database 4-2

c

c_variable 6-5,7-4,7-7
char 11-4
clear 7-2,D-2
<CMDFILE> B-8
<CMDFILE> softkey 3-2
command files 9-3,10-2
comments displayed as source lines E-6
comp_db 3-4,4-3,6-1
comp_db file not current E-3
comp_db files 4-1
comp_db option 3-3
comp_sym 3-4,4-3,6-1

comp_sym option 3-3
compiler directives 4-1
compiler overhead code 11-1
compiler symbol file 3-14
compiling files 4-3
configuration D-2
configuration files 3-14,9-2
configure load_from 9-3
configure save_in 9-2
configure softkey 3-2
configuring the analyzer 9-1
copy D-2
copy command 8-8
copy softkey 3-2,10-3
current line 8-3

d

data missed on recursive call exit E-2
data types 1-5,11-1
data_flow D-3
database file 3-4,4-4
database_check D-3
database_check command syntax diagram 4-6
default 8-6,D-3
default path 6-1,11-2
default_path 3-12,D-3
defining a default path 3-12
defining software breakpoint vector 3-5
DELETE CHAR key 10-5
depth D-3
<DEPTH> B-8
development phase 1-2
directed syntax 10-1
display 1-7,7-1,8-8,D-3
display command 7-4,8-6
display command 7-4,8-6
display softkey 3-1
displaying pad bytes 8-4
displaying pad bytes 8-4
double 11-4
duplicating floppy discs 2-1
dynamic symbols 11-1,11-3
dynamically stored symbols 4-1

e

emulation bus analyzer 1-1
emulation command file 3-8,5-1
emulation command file? 5-1
emulation commands 5-3
emulation configuration 5-1

emulation configuration file 5-1
emulation control 1-7
emulation interface 5-1
emulation memory subsystem 1-1
emulation subsystem 1-1
emulation system 2-1
emulation_memory 5-5,D-4
emulator analysis mode, selecting 3-8,5-2
emulator status message not updated E-4
end D-4
end softkey 3-2,10-4
end statement displayed at beginning of procedure E-6
entering module/variable names 10-2
entering numeric values 10-2
entry 11-1
entry point 3-14
enum 11-4
error message ': XXX is not found' is displayed E-5
error message 'access to guarded memory' displayed E-5
error message 'access to guarded memory' displayed E-5
error message 'bad line range' displayed E-5
error message 'program execution outside of E-6
error message 'symbol not found' displayed E-6
error occurs on execution of measurement E-4
executable code 11-2
execute D-4
execute softkey 3-2,10-3
execution environment 1-2,4-1
exit 11-1
expected go to not displayed E-1
extra accesses to a record displayed E-7
extra index to an array field displayed E-7

f

field and display width 8-5
file 11-2,D-5
<FILE> B-8
file names 10-3
fixed parameters 4-7
float 11-4
from D-5
function return values not displayed E-3
functions 11-1

g

general user information 1-9
generate_database command syntax 4-5
generate_database command, examples 4-5
generate_database command, parameters 4-5
generate_database utility 3-4

generate_database utility, execution 3-5
generate_database utility, required files 3-5,4-4
getting started 3-1
global overview 1-2
global variables 11-1

h

halt D-5

i

IEEE simple precision 11-4
illegal values 8-5
incomplete access to variables 8-6,E-7
incorrect values displayed E-8
<INDEX> 6-5,6-16,7-5,7-7,B-8
initial turn on 3-3
INSERT CHAR key 10-5
installation 2-1
int 11-4
integer 11-4
interactive capabilities 1-7
intermodule bus 6-3
internal analyzer 6-3
interpreting the trace listing 3-14
intrinsic data types 11-3
<INVALID> B-9

l

labels 11-2
<LINE> 5-9,6-12,7-3,B-9
line numbers 11-2
link time 11-1
link_sym file 3-4,4-3
linking 3-2
linking files 4-3
listfile D-5
load 1-7,D-5
load command 5-5
load softkey 3-2
load_from D-5
loading a measurement configuration 9-3
loading and executing a program in emulation 3-8
loading and running programs 3-10,5-1
local variable 6-3
local variable cannot be modified E-4
local variables 1-3,11-1,11-3
logging commands 10-3
long 11-4

longreal 11-4
lower case module/variable names 10-2
lower case module/variable names 10-2

m

major softkey levels 3-1
making sequential trace measurements 6-2
manual applicability 1-1
measurement 8-8
measurement capabilities 1-2
measurement configuration 9-1
measurement considerations 6-2
measurement modes 6-1
measurement not completing E-7
measurement system interface 6-3
member 11-5
missing data during recursive calls E-2
missing symbols on the display E-2
  accesses to variables not displayed E-3
  array parameter in C not displayed E-2
  data missed on recursive call exit E-2
  function return values not displayed E-3
  object of pointer not displayed E-2
  object of pointer not displayed E-2
  reads or writes to a variable are missing E-3
  reads or writes to a variable are missing E-3
missing source statements E-1
  containing 'end' or '}' not displayed E-1
  expected go to not displayed E-1
  missing data during recursive calls E-2
  source line missing for entry of first procedure in a file E-2
  statement containing 'end' or '}' not displayed E-1
  variable accesses in statement E-1
modify 1-7,7-1,D-6
modify command 7-6
modify softkey 3-1
<MODULE> 6-5,6-9,6-12,7-3,B-9
module characteristics 11-1
module vs. proc 11-1
module/variable names, lower case 10-2
modules 11-1,D-6
modules/variable names, entering 10-2

n

number of acquired states greater than specified E-3
number of acquired states greater than specified E-3

o

object of pointer not displayed . . . E-2
<OFFSET> . . . 5-9
on_entry . . . 6-5,7-3,D-6
on_exit . . . 6-5,7-3,D-6
operating syntax diagrams . . . A-1
OPTIMIZE . . . 4-7
options continue . . . 9-1
or . . . D-7

p

p_variable . . . 6-6,7-5,7-7
padded fields . . . 11-6
padding . . . 11-6
<PARMS> . . . B-9
partial values displayed . . . E-8
paths . . . 11-2
(PC=nnnnH) displayed . . . E-5
performing a trace modules measurement . . . 3-10
pointer . . . 11-5
prefetch . . . 6-7,6-9,6-13
preparing the system for measurements . . . 3-2
printer . . . 8-9,D-7
proc . . . 11-2,D-7
<PROC> . . . 6-5,6-16,7-5,7-7,B-9
procedures . . . 11-1
program activity overview . . . 3-14
program counter . . . 5-2
program crashes . . . E-4
program variables . . . 1-3
programs . . . 11-1
prompt softkeys . . . 10-5

r

read . . . 6-16,D-8
reads or writes to a variable are missing . . . E-3
real . . . 11-4
RECALL key . . . 10-5
recommended programming style . . . 3-16
record . . . 11-5
recursive . . . 3-14
reference manual updates . . . 1-8
reference parameters . . . 11-3
register variable not displayed . . . E-2
reset . . . 1-7,D-8
reset . . . 1-7,D-8
reset softkey . . . 3-2
<RETURN> . . . B-9
run . . . 1-7,D-8

run command 5-8
run softkey 3-2
run time 11-1

S

save_in D-8
saving the configuration 3-14
scalar data types 11-1
selecting the emulation analysis mode 3-8,5-2
set 11-5
setting up the trace specification 3-13
setup D-8
setup display 3-13
setup softkey 3-1
short 11-3
show D-9
show command 8-9
show softkey 3-2,10-3
SIGNED_8 11-3
SIGNED_16 11-4
SIGNED_16 11-4
simulated I/O 6-3
software breakpoint vector, defining 3-5
software materials subscription 1-8
software problem reporting 1-8
software release bulletins 1-8
software status bulletins 1-8
software updates 1-8
source 8-6,D-9
source field 8-1
source line is not line associated with symbol E-6
source path field 8-2
source_path 8-6
special values 8-5
specifying the acquisition depth 3-13
src_path D-9
start D-9
start-up vector 5-2
<STATE #> 8-10,B-10
statement containing 'end' or '}' not displayed E-1
statements D-9
static variables 6-3,11-1
statically stored symbols 4-1
status 1-5,8-6,D-10
status field 8-2
storage classes 11-1
structure 11-5
structured data types 11-1,11-5
SUBRANGE TYPE 11-4
sw_anly D-10
symbol 8-7,D-10
<SYMBOL> 5-9,B-10

symbol field 8-2
symbol path field 8-2
symbol_path 8-7,D-10
symbolic data base 3-10
symbolic data types 11-3
symbolic interface 4-1
symbolic interface diagram 4-2
symbols 11-1

t

TAB key 10-5
then 8-7,D-10
thru 8-9,D-11
to D-11
trace. D-11
trace data_flow 1-3,6-3
trace file 6-1
trace measurements. 1-2
trace modules 1-4
trace modules measurement 6-8
trace or run command does not work correctly E-4
trace statements. 1-5
trace statements measurement 6-11
trace variables 1-6
trace variables measurement 6-14
transfer. D-11
transfer address 5-2

u

unexpected analyzer execution E-3
  analyzer does not capture data E-3
  analyzer does not find breakpoint E-3
  analyzer does not find breakpoint E-3
  local variable cannot be modified E-4
  number of acquired states greater than specified E-3
  variables cannot be modified in their native type E-4
unexpected emulation operation E-4
  error occurs on execution of measurement. E-4
  trace or run command does not work correctly. E-4
unexpected error or status message E-4
  (PC=nnnnH)' displayed E-5
  absolute file (PC=nnnnH)' displayed E-6
  absolute file loaded without emulator being reset E-5
  analyzer does not accept configuration file E-5
  apparently correct command does not work E-5
  emulator status message not updated. E-4
  error message ': XXX is not found' is displayed. E-5
  error message 'access to guarded memory' displayed E-5
  error message 'access to guarded memory' displayed E-5
  error message 'bad line range' displayed E-5

error message 'program execution outside of E-6
error message 'symbol not found' displayed E-6
error message 'symbol not found' displayed E-6
unexpected read/write operations E-7
unexpected symbols on the display E-7
extra accesses to a record displayed E-7
extra index to an array field displayed E-7
unexpected read/write operations E-7
variable name different from expected name E-7
unexpected source line E-6
'end' statement displayed at beginning of procedure E-6
comments displayed as source lines E-6
source line is not line associated with symbol E-6
unexpected termination of measurement E-7
unexpected values displayed
incomplete access to variables E-7
incorrect values displayed E-8
incorrect values displayed E-8
value of symbol is illegal E-7
unions 11-5
unsigned 11-4
unsigned long 11-4
UNSIGNED_8 11-4
UNSIGNED_16 11-4
UNSIGNED_32 11-4
user vs. supervisor memory space, HP 64234/HP 64244/HP 64245 5-2
user-definable 11-4
user_mem D-12
user_memory 5-6
userid 3-3,10-1
using compiler directives 4-7
using compiler directives 4-7
using the emulation monitor 5-3
using the generate_database command 4-4
utility commands 10-3
utility keys used for transportation-t 3-10

V

<VALUE> 7-7,B-10
value field 8-2
value of symbol is illegal E-7
value parameters 6-3,11-3
<VAR> 6-6,6-167-5,7-7,B-10
variable accesses in statement E-1
variable name different from expected name E-7
variable/module names, entering 10-2
variable/module names, lower case 10-2
variables 6-16,D-12
variables cannot be modified in their native type E-4
variant records 11-5
viewing measurement data 8-1

W

wait D-12
wait softkey 10-4
width 8-7
<WIDTH> B-10
write. 6-16,D-12
write_protected 9-2,D-7

FOLD HERE

HEWLETT PACKARD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 1303 COLORADO SPRINGS, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD
LOGIC PRODUCT SUPPORT DEPT.
Attn: Technical Publications Manager
Centennial Annex — D2
P.O. Box 617
Colorado Springs, Colorado 80901-0617

FOLD HERE

Your cooperation in completing and returning this form will be greatly appreciated. Thank you.

READER COMMENT SHEET

Operating Manual, Model 64330 Series
High Level Software Analyzers 68000/68008/68010
64331-90903 E0985, September 1985

Your comments are important to us. Please answer this questionnaire and return it to us. Circle the number that best describes your answer in questions 1 through 7. Thank you.

1. The information in this book is complete:

Doesn't cover enough (what more do you need?)	1 2 3 4 5	Covers everything

2. The information in this book is accurate:

Too many errors	1 2 3 4 5	Exactly right

3. The information in this book is:

Difficult to find	1 2 3 4 5	Easy to find

4. The Index and Table of Contents are useful:

Missing or inadequate	1 2 3 4 5	Helpful

5. What about the "how-to" procedures and examples:

No help	1 2 3 4 5	Very Helpful
Not enough	1 2 3 4 5	Too many

6. What about the writing style:

Confusing	1 2 3 4 5	Clear

7. What about organization of the book:

Poor order	1 2 3 4 5	Good order

8. What about the size of the book:

Too small	1 2 3 4 5	Too big

Comments: ____

____
____
____
____

Particular pages with errors? ______________________________
Name: ______________________________
Job title: ______________________________
Company: ______________________________
Address: ______________________________

Note: If mailed outside U.S.A., place card in envelope. Use address shown on other side of this card.

FOLD HERE

HEWLETT PACKARD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 1303 COLORADO SPRINGS, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD
LOGIC PRODUCT SUPPORT DEPT.
Attn: Technical Publications Manager
Centennial Annex – D2
P.O. Box 617
Colorado Springs, Colorado 80901-0617

FOLD HERE

Your cooperation in completing and returning this form
will be greatly appreciated. Thank you.

READER COMMENT SHEET

Operating Manual, Model 64330 Series
High Level Software Analyzers 68000/68008/68010
64331-90903 E0985, September 1985

Your comments are important to us. Please answer this questionnaire and return it to us. Circle the number that best describes your answer in questions 1 through 7. Thank you.

1. The information in this book is complete:

Doesn't cover enough (what more do you need?)	1 2 3 4 5	Covers everything

2. The information in this book is accurate:

Too many errors	1 2 3 4 5	Exactly right

3. The information in this book is:

Difficult to find	1 2 3 4 5	Easy to find

4. The Index and Table of Contents are useful:

Missing or inadequate	1 2 3 4 5	Helpful

5. What about the "how-to" procedures and examples:

No help	1 2 3 4 5	Very Helpful
Not enough	1 2 3 4 5	Too many

6. What about the writing style:

Confusing	1 2 3 4 5	Clear

7. What about organization of the book:

Poor order	1 2 3 4 5	Good order

8. What about the size of the book:

Too small	1 2 3 4 5	Too big

Comments: ______________________________

____

____

____

____

Particular pages with errors? ______________________________
Name: ______________________________
Job title: ______________________________
Company: ______________________________
Address: ______________________________

Note: If mailed outside U.S.A., place card in envelope. Use address shown on other side of this card.

SALES & SUPPORT OFFICES
Arranged alphabetically by country

Product Line Sales/Support Key

Key Product Line
A Analytical
CM Components
C Computer Systems
E Electronic Instruments & Measurement Systems
M Medical Products
P Personal Computation Products
*** Sales only for specific product line**
**** Support only for specific product line**

IMPORTANT:These symbols designate general product line capability.They do not insure sales or support availability for all products within a line, at all locations.Contact your local sales office for information regarding locations where HP support is available for specific products.

HEADQUARTERS OFFICES
If there is no sales office listed for your area, contact one of these headquarters offices.

NORTH/CENTRAL AFRICA
Hewlett-Packard S.A.
7, rue du Bois-du-Lan
CH-1217 **MEYRIN** 1, Switzerland
Tel: (022) 83 12 12
Telex: 27835 hmea
Cable: HEWPACKSA Geneve

ASIA
Hewlett-Packard Asia Ltd.
47/F, 26 Harbour Rd.,
Wanchai, **HONG KONG**
G.P.O. Box 863, Hong Kong
Tel: 5-8330833
Telex: 76793 HPA HX
Cable: HPASIAL TD

CANADA
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
MISSISSAUGA, Ontario L4V 1M8
Tel: (416) 678-9430
Telex: 610-492-4246

EASTERN EUROPE
Hewlett-Packard Ges.m.b.h.
Lieblgasse 1
P.O.Box 72
A-1222 **VIENNA,** Austria
Tel: (222) 2500-0
Telex: 1 3 4425 HEPA A

NORTHERN EUROPE
Hewlett-Packard S.A.
Uilenstede 475
P.O.Box 999
NL-1183 AG **AMSTELVEEN**
The Netherlands
Tel: 20 437771
Telex: 18 919 hpner nl

SOUTH EAST EUROPE
Hewlett-Packard S.A.
World Trade Center
110 Avenue Louis Casai
1215 Cointrin, **GENEVA,** Switzerland
Tel: (022) 98 96 51
Telex: 27225 hpser

MEDITERRANEAN AND MIDDLE EAST
Hewlett-Packard S.A.
Mediterranean and Middle East Operations
Atrina Centre
32 Kifissias Ave.
Paradissos-Amarousion, **ATHENS**
Greece
Tel: 682 88 11
Telex: 21-6588 HPAT GR
Cable: HEWPACKSA Athens

UNITED KINGDOM
Hewlett-Packard Ltd.
Nine Mile Ride
Easthampstead, **WOKINGHAM**
Berkshire, IRGII 3LL
Tel: 0344 773100
Telex: 848805

EASTERN USA
Hewlett-Packard Co.
4 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 258-2000

MIDWESTERN USA
Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800

SOUTHERN USA
Hewlett-Packard Co.
2000 South Park Place
P.O. Box 105005
ATLANTA, GA 30348
Tel: (404) 955-1500

WESTERN USA
Hewlett-Packard Co.
3939 Lankershim Blvd.
P.O. Box 3919
LOS ANGELES, CA 91604
Tel: (213) 506-3700

OTHER INTERNATIONAL AREAS
Hewlett-Packard Co.
Intercontinental Headquarters
3495 Deer Creek Road
PALO ALTO, CA 94304
Tel: (415) 857-1501
Telex: 034-8300
Cable: HEWPACK

ANGOLA
Telectra Angola LDA
Empresa Tecnica de Equipamentos
Rua Conselheiro Julio de Vilhema, 16
Caixa Postal 6487
LUANDA
Tel: 35515,35516
Telex: 3134
E,C*

ARGENTINA
Hewlett-Packard Argentina S.A.
Montaneses 2140/50
1428 **BUENOS AIRES**
Tel: 783-4886/4836/4730
Cable: HEWPACKARG
A,C,CM,E,P

Biotron S.A.C.I.e.I.
Av. Paso Colon 221, Piso 9
1399 **BUENOS AIRES**
CM

Laboratorio Rodriguez
Corswant S.R.L.
Misiones, 1156 - 1876
Bernal, Oeste
BUENOS AIRES
Tel: 252-3958, 252-4991
A

Argentina Esanco S.R.L.
Avasco 2328
1416 **BUENOS AIRES**
Tel: 541-58-1981, 541-59-2767
A

AUSTRALIA

Adelaide, South Australia Office
Hewlett-Packard Australia Ltd.
153 Greenhill Road
PARKSIDE, S.A. 5063
Tel: 272-5911
Telex: 82536
Cable: HEWPARD Adelaide
A*,C,CM,E,M,P

Brisbane, Queensland Office
Hewlett-Packard Australia Ltd.
10 Payne Road
THE GAP, Queensland 4061
Tel: 30-4133
Telex: 42133
Cable: HEWPARD Brisbane
A,C,CM,E,M,P

Canberra, Australia Capital Territory Office
Hewlett-Packard Australia Ltd.
121 Wollongong Street
FYSHWICK, A.C.T. 2609
Tel: 80 4244
Telex: 62650
Cable: HEWPARD Canberra
C,CM,E,P

Melbourne, Victoria Office
Hewlett-Packard Australia Ltd.
31-41 Joseph Street
BLACKBURN, Victoria 3130
Tel: 895-2895
Telex: 31-024
Cable: HEWPARD Melbourne
A,C,CM,E,M,P

Perth, Western Australia Office
Hewlett-Packard Australia Ltd.
261 Stirling Highway
CLAREMONT, W.A. 6010
Tel: 383-2188
Telex: 93859
Cable: HEWPARD Perth
A,C,CM,E,M,P

Sydney, New South Wales Office
Hewlett-Packard Australia Ltd.
17-23 Talavera Road
P.O. Box 308
NORTH RYDE, N.S.W. 2113
Tel: 888-4444
Telex: 21561
Cable: HEWPARD Sydney
A,C,CM,E,M,P

AUSTRIA
Hewlett-Packard Ges.m.b.h.
Verkauisbüro Graz
Grottenhofstrasse 94
A-8052 **GRAZ**
Tel: (0316) 291 56 60
Telex: 32375
C,E

Hewlett-Packard Ges.m.b.h.
Lieblgasse 1
P.O. Box 72
A-1222 **VIENNA**
Tel: (0222) 2500-0
Telex: 134425 HEPA A
A,C,CM,E,M,P

BAHRAIN
Green Salon
P.O. Box 557
MANAMA
Tel: 255503-255950
Telex: 8441
P

Wael Pharmacy
P.O. Box 648
MANAMA
Tel: 256123
Telex: 8550 WAEL BN
E,M

Zayani Computer Systems
218 Shaik Mubarak Building
Government Avenue
P.O. Box 5918
MANAMA
Tel: 276278
Telex: 9015
P

BELGIUM
Hewlett-Packard Belgium S.A./N.V.
Blvd de la Woluwe, 100
Woluwedal
B-1200 **BRUSSELS**
Tel: (02) 762-32-00
Telex: 23-494 paloben bru
A,C,CM,E,M,P

BERMUDA
Applied Computer Technologies
Atlantic House Building
Par-La-Ville Road
Hamilton 5
Tel: 295-1616
P

BRAZIL
Hewlett-Packard do Brasil
I.e.C. Ltda.
Alameda Rio Negro, 750
Alphaville
06400 **BARUERI SP**
Tel: (011) 421.1311
Telex: (011) 33872 HPBR-BR
Cable: HEWPACK Sao Paulo
A,C,CM,E,M,P

Hewlett-Packard do Brasil
I.e.C. Ltda.
Praia de Botafago 228
6° Andar-conj 614
Edificio Argentina - Ala A
22250 **RIO DE JANEIRO**
Tel: (02l) 552-6422
Telex: 21905 HPBR-BR
Cable: HEWPACK Rio de Janeiro
A,C,CM,E,P*

Convex/Van Den
Rua Jose Bonifacio
458 Todos Os Santos
CEP 20771
RIO DE JANEIRO, RJ
Tel: 591-0197
Telex: 33487 EGLB BR
A

ANAMED I.C.E.I. Ltda.
Rua Bage, 103
04012 **SAO PAULO,** SP
Tel: (011) 572-6537
Telex: 24720 HPBR-BR
M

Datatronix Electronica Ltda.
Av. Pacaembu 746-C11
SAO PAULO, SP
Tel: (118) 260111
CM

CAMEROON
Beriac
B. P. 23
DOUALA
Tel: 420153
Telex: 5351
C,P

CANADA

Alberta
Hewlett-Packard (Canada) Ltd.
3030 3rd Avenue N.E.
CALGARY, Alberta T2A 6T7
Tel: (403) 235-3100
A,C,CM,E*,M,P*

Hewlett-Packard (Canada) Ltd.
11120-178th Street
EDMONTON, Alberta T5S 1P2
Tel: (403) 486-6666
A,C,CM,E,M,P

SALES & SUPPORT OFFICES
Arranged alphabetically by country

CANADA (Cont'd)

British Columbia
Hewlett-Packard (Canada) Ltd.
10691 Shellbridge Way
RICHMOND,
British Columbia V6X 2W7
Tel: (604) 270-2277
Telex: 610-922-5059
A,C,CM,E*,M,P*

Hewlett-Packard (Canada) Ltd.
121 - 3350 Douglas Street
VICTORIA, British Columbia V8Z 3L1
Tel: (604) 381-6616
C

Manitoba
Hewlett-Packard (Canada) Ltd.
1825 Inkster Blvd.
WINNIPEG, Manitoba R2X 1R3
Tel: (204) 694-2777
A,C,CM,E,M,P*

New Brunswick
Hewlett-Packard (Canada) Ltd.
814 Main Street
MONCTON, New Brunswick E1C 1E6
Tel: (506) 855-2841
C

Nova Scotia
Hewlett-Packard (Canada) Ltd.
Suite 111
900 Windmill Road
DARTMOUTH, Nova Scotia B3B 1P7
Tel: (902) 469-7820
C,CM,E*,M,P*

Ontario
Hewlett-Packard (Canada) Ltd.
3325 N. Service Rd., Unit 3
BURLINGTON, Ontario L7N 3G2
Tel: (416) 335-8644
C,M*

Hewlett-Packard (Canada) Ltd.
496 Days Road
KINGSTON, Ontario K7M 5R4
Tel: (613) 384-2088
C

Hewlett-Packard (Canada) Ltd.
552 Newbold Street
LONDON, Ontario N6E 2S5
Tel: (519) 686-9181
A,C,CM,E*,M,P*

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
MISSISSAUGA, Ontario L4V 1M8
Tel: (416) 678-9430
A,C,CM,E,M,P

Hewlett-Packard (Canada) Ltd.
2670 Queensview Dr.
OTTAWA, Ontario K2B 8K1
Tel: (613) 820-6483
A,C,CM,E*,M,P*

Hewlett-Packard (Canada) Ltd.
The Oaks Plaza, Unit #9
2140 Regent Street
SUDBURY, Ontario, P3E 5S8
Tel: (705) 522-0202
C

Hewlett-Packard (Canada) Ltd.
3790 Victoria Park Ave.
WILLOWDALE, Ontario M2H 3H7
Tel: (416) 499-2550
C

Quebec
Hewlett-Packard (Canada) Ltd.
17500 Trans Canada Highway
South Service Road
KIRKLAND, Quebec H9J 2X8
Tel: (514) 697-4232
A,C,CM,E,M,P*

Hewlett-Packard (Canada) Ltd.
1150 rue Claire Fontaine
QUEBEC CITY, Quebec G1R 5G4
Tel: (418) 648-0726
C

Hewlett-Packard (Canada) Ltd.
130 Robin Crescent
SASKATOON, Saskatchewan S7L 6M7
Tel: (306) 242-3702
C

CHILE
ASC Ltda.
Austria 2041
SANTIAGO
Tel: 223-5946, 223-6148
Telex: 340192 ASC CK
C,P

Isical Ltda.
Av. Italia 634 Santiago
Casilla 16475
SANTIAGO 9
Tel: 222-0222
Telex: 440283 JCYCL CZ
CM,E,M

Metrolab S.A.
Monjitas 454 of. 206
SANTIAGO
Tel: 395752, 398296
Telex: 340866 METLAB CK
A

Olympia (Chile) Ltda.
Av. Rodrigo de Araya 1045
Casilla 256-V
SANTIAGO 21
Tel: 225-5044
Telex: 340892 OLYMP
Cable: Olympiachile Santiagochile
C,P

CHINA, People's Republic of
China Hewlett-Packard, Ltd.
47/F China Resources Bldg.
26 Harbour Road
HONG KONG
Tel: 5-8330833
Telex: 76793 HPA HX
Cable: HP ASIA LTD
A*,M*

China Hewlett-Packard, Ltd.
P.O. Box 9610, Beijing
4th Floor, 2nd Watch Factory Main Bldg.
Shuang Yu Shu, Bei San Huan Rd.
Hai Dian District
BEIJING
Tel: 28-0567
Telex: 22601 CTSHP CN
Cable: 1920 Beijing
A,C,CM,E,M,P

COLOMBIA
Instrumentación
H. A. Langebaek & Kier S.A.
Carrera 4A No. 52A-26
Apartado Aereo 6287
BOGOTA 1, D.E.
Tel: 212-1466
Telex: 44400 INST CO
Cable: AARIS Bogota
CM,E,M

Nefromedicas Ltda.
Calle 123 No. 9B-31
Apartado Aereo 100-958
BOGOTA D.E., 10
Tel: 213-5267, 213-1615
Telex: 43415 HEGAS CO
A

Compumundo
Avenida 15 # 107-80
BOGOTA D.E.
Tel: 214-4458
Telex: 45466 MARICO
P

Carvajal, S.A.
Calle 29 Norte No. 6A-40
Apartado Aereo 46
CALI
Tel: 368-1111
Telex: 55650
C,E,P

CONGO
Seric-Congo
B. P. 2105
BRAZZAVILLE
Tel: 815034
Telex: 5262

COSTA RICA
Cientifica Costarricense S.A.
Avenida 2, Calle 5
San Pedro de Montes de Oca
Apartado 10159
SAN JOSÉ
Tel: 24-38-20, 24-08-19
Telex: 2367 GALGUR CR
CM,E,M

CYPRUS
Telerexa Ltd.
P.O. Box 4809
14C Stassinos Avenue
NICOSIA
Tel: 62698
Telex: 2894 LEVIDO CY
E,M,P

DENMARK
Hewlett-Packard A/S
Datavej 52
DK-3460 **BIRKEROD**
Tel: (02) 81-66-40
Telex: 37409 hpas dk
A,C,CM,E,M,P

Hewlett-Packard A/S
Rolighedsvej 32
DK-8240 **RISSKOV,** Aarhus
Tel: (06) 17-60-00
Telex: 37409 hpas dk
C,E

DOMINICAN REPUBLIC
Microprog S.A.
Juan Tomás Mejía y Cotes No. 60
Arroyo Hondo
SANTO DOMINGO
Tel: 565-6268
Telex: 4510 ARENTA DR (RCA)
P

ECUADOR
CYEDE Cia. Ltda.
Avenida Eloy Alfaro 1749
y Belgica
Casilla 6423 CCI
QUITO
Tel: 450-975, 243-052
Telex: 22548 CYEDE ED
CM,E,P

Medtronics
Valladolid 524 Madrid
P.O. 9171, **QUITO**
Tel: 223-8951
Telex: 2298 ECKAME ED
A

Hospitalar S.A.
Robles 625
Casilla 3590
QUITO
Tel: 545-250, 545-122
Telex: 2485 HOSPTL ED
Cable: HOSPITALAR-Quito
M

Ecuador Overseas Agencies C.A.
Calle 9 de Octubre #818
P.O. Box 1296, Guayaquil
QUITO
Tel: 306022
Telex: 3361 PBCGYE ED
M

EGYPT
Sakrco Enterprises
70, Mossadak Str.
Dokki, Giza
CAIRO
Tel: 706440
Telex: 93146
C

International Engineering Associates
24 Hussein Hegazi Street
Kasr-el-Ain
CAIRO
Tel: 23829, 21641
Telex: 93830 IEA UN
Cable: INTEGASSO
E,M*

S.S.C. Medical
40 Gezerat El Arab Street
Mohandessin
CAIRO
Tel: 803844, 805998, 810263
Telex: 20503 SSC UN
M*

EL SALVADOR
IPESA de El Salvador S.A.
29 Avenida Norte 1223
SAN SALVADOR
Tel: 26-6858, 26-6868
Telex: 20539 IPESA SAL
A,C,CM,E,P

ETHIOPIA
Seric-Ethiopia
P.O. Box 2764
ADDIS ABABA
Tel: 185114
Telex: 21150
C,P

FINLAND
Hewlett-Packard Oy
Piispankalliontie 17
02200 **ESPOO**
Tel: 00358-0-88721
Telex: 121563 HEWPA SF
A,C,CM,E,M,P

FRANCE
Hewlett-Packard France
Z.I. Mercure B
Rue Berthelot
13763 Les Milles Cedex
AIX-EN-PROVENCE
Tel: (42) 59-41-02
Telex: 410770F
A,C,E,M,P*

Hewlett-Packard France
64, rue Marchand Saillant
61000 **ALENCON**
Tel: (33) 29 04 42

Hewlett-Packard France
28 rue de la Republique
Boite Postale 503
25026 **BESANCON** Cedex
Tel: (81) 83-16-22
Telex: 361157
C,M

Hewlett-Packard France
Chemin des Mouilles
Boite Postale 162
69130 **ECULLY** Cedex (Lyon)
Tel: (78) 833-81-25
Telex: 310617F
A,C,E,M

Hewlett-Packard France
Parc d'activités du Bois Briard
2, avenue du Lac
91040 **EVRY** Cedex
Tel: 6 077-96 60
Telex: 692315F
E

Hewlett-Packard France
5, avenue Raymond Chanas
38320 **EYBENS** (Grenoble)
Tel: (76) 62-57-98
Telex: 980124 HP GRENOB EYBE
C

Hewlett-Packard France
Rue Fernand. Forest
Z.A. Kergaradec
29239 **GOUESNOU**
Tel: (98) 41-87-90

Hewlett-Packard France
Centre d'affaires Paris-Nord
Bâtiment Ampère
Rue de la Commune de Paris
Boite Postale 300
93153 **LE BLANC-MESNIL**
Tel: (1) 865-44-52
Telex: 211032F
C,E,M

Hewlett-Packard France
Parc d'activités Cadera
Quartier Jean-Mermoz
Avenue du Président JF Kennedy
F-33700 **MÉRIGNAC** (Bordeaux)
Tel: (56) 34-00-84
Telex: 550105F
C,E,M

Hewlett-Packard France
Immeuble "Les 3 B"
Nouveau chemin de la Garde
ZAC du Bois Briand
44085 **NANTES** Cedex
Tel: (40) 50-32-22
Telex: 711085F
C**

Hewlett-Packard France
125, rue du Faubourg Bannier
45000 **ORLEANS**
Tel: (38) 68 01 63

FRANCE (Cont'd)

Hewlett-Packard France
Zone Industrielle de Courtaboeuf
Avenue des Tropiques
91947 Les Ulis Cedex **ORSAY**
Tel: (6) 907-78-25
Telex: 600048F
A,C,CM,E,M,P

Hewlett-Packard France
Paris Porte-Maillot
15, boulevard de L'Amiral-Bruix
75782 **PARIS** Cedex 16
Tel: (1) 502-12-20
Telex: 613663F
C,M,P

Hewlett-Packard France
124, Boulevard Tourasse
64000 **PAU**
Tel: (59) 80 38 02

Hewlett-Packard France
2 Allée de la Bourgonnette
35100 **RENNES**
Tel: (99) 51-42-44
Telex: 740912F
C,CM,E,M,P*

Hewlett-Packard France
98 avenue de Bretagne
76100 **ROUEN**
Tel: (35) 63-57-66
Telex: 770035F
C

Hewlett-Packard France
4, rue Thomas-Mann
Boite Postale 56
67033 **STRASBOURG** Cedex
Tel: (88) 28-56-46
Telex: 890141F
C,E,M,P*

Hewlett-Packard France
La Péripole III
20, chemin du Pigeonnier de la Cépière
F-31083 **TOULOUSE** Cedex
Tel: (61) 40-11-12
Telex: 531639F
A,C,E,P*

Hewlett-Packard France
9, rue Baudin
26000 **VALENCE**
Tel: (75) 42 76 16

Hewlett-Packard France
Carolor
ZAC de Bois Briand
57640 **VIGY** (Metz)
Tel: (8) 771 20 22
C

Hewlett-Packard France
Parc d'activité des Prés
1, rue Papin
59658 **VILLENEUVE D'ASCQ** Cedex
Tel: (20) 47 78 78
Telex: 160124F
C,E,M,P*

GABON

Sho Gabon
P.O. Box 89
LIBREVILLE
Tel: 721 484
Telex: 5230

GERMAN FEDERAL REPUBLIC

Hewlett-Packard GmbH
Geschäftsstelle
Keithstrasse 2-4
D-1000 **BERLIN** 30
Tel: (030) 21 99 04-0
Telex: 018 3405 hpbln d
A,C,E,M,P

Hewlett-Packard GmbH
Vertriebszentrun Südwest
Schickardstrasse 2
D-7030 **BÖBLINGEN**
Tel: (07031) 645-0
Telex: 7265 743 hep
A,C,CM,E,M,P

Hewlett-Packard GmbH
Vertriebszentrum West
Berliner Strasse III
D-4030 **RATINGEN** 3
Tel: (02102) 494-0
Telex: 589 070 hprad
A,C,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Schleefstr. 28a
D-4600 **DORTMUND**-41
Tel: (0231) 45001
Telex: 822858 hepdad
A,C,E

Hewlett-Packard GmbH
Vertriebszentrum Mitte
Hewlett-Packard-Strasse
D-6380 **BAD HOMBURG**
Tel: (06172) 400-0
Telex: 410 844 hpbhg
A,C,E,M,P

Hewlett-Packard GmbH
Vertriebszentrum Nord
Kapstadtring 5
D-2000 **HAMBURG** 60
Tel: (040) 63804-1
Telex: 021 63 032 hphh d
A,C,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Heidering 37-39
D-3000 **HANNOVER** 61
Tel: (0511) 5706-0
Telex: 092 3259
A,C,CM,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Rosslauer Weg 2-4
D-6800 **MANNHEIM**
Tel: (0621) 70 05-0
Telex: 0462105
A,C,E

Hewlett-Packard GmbH
Geschäftsstelle
Messerschmittstrasse 7
D-7910 **NEU ULM**
Tel: (0731) 70 73-0
Telex: 0712816 HP ULM-D
A,C,E*

Hewlett-Packard GmbH
Geschäftsstelle
Emmericher Strasse 13
D-8500 **NÜRNBERG** 10
Tel: (0911) 5205-0
Telex: 0623 860 hpnbg
C,CM,E,M,P

Hewlett-Packard GmbH
Vertriebszentrum Süd
Eschenstrasse 5
D-8028 **TAUFKIRCHEN**
Tel: (089) 61 20 7-0
Telex: 0524985
A,C,CM,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Ermlisallee
7517 **WALDBRONN** 2
Tel: (07243) 602-0
Telex: 782 838 hepk
A,C,E

GREAT BRITAIN

See United Kingdom

GREECE

Hewlett-Packard A.E.
178, Kifissias Avenue
6th Floor
Halandri-**ATHENS**
Greece
Tel: 6471543, 6471673, 6472971
Telex: 221 286 HPHLGR
A,C,CM**,E,M,P

Kostas Karaynnis S.A.
8, Omirou Street
ATHENS 133
Tel: 32 30 303, 32 37 371
Telex: 215962 RKAR GR
A,C*,CM,E

Impexin
Intelect Div.
209 Mesogion
11525 **ATHENS**
Tel: 6474481/2
Telex: 216286
P

Haril Company
38, Mihalakopoulou
ATHENS 612
Tel: 7236071
Telex: 218767
M*

Hellamco
P.O. Box 87528
18507 **PIRAEUS**
Tel: 4827049
Telex: 241441
A

GUATEMALA

IPESA
Avenida Reforma 3-48, Zona 9
GUATEMALA CITY
Tel: 316627, 314786
Telex: 3055765 IPESA GU
A,C,CM,E,M,P

HONG KONG

Hewlett-Packard Hong Kong, Ltd.
G.P.O. Box 795
5th Floor, Sun Hung Kai Centre
30 Harbour Road
HONG KONG
Tel: 5-8323211
Telex: 66678 HEWPA HX
Cable: HEWPACK Hong Kong
E,C,P

CET Ltd.
10th Floor, Hua Asia Bldg.
64-66 Gloulester Road
HONG KONG
Tel: (5) 200922
Telex: 85148 CET HX
CM

Schmidt & Co. (Hong Kong) Ltd.
18th Floor, Great Eagle Centre
23 Harbour Road
HONG KONG
Tel: 5-8330222
Telex: 74766 SCHMC HX
A,M

ICELAND

Hewlett-Packard Iceland
Hoefdabakka 9
110 **Reykjavik**
Tel: (1) 67 1000
A,C,CM,E,M,P

INDIA

Computer products are sold through Blue Star Ltd.All computer repairs and maintenance service is done through Computer Maintenance Corp.

Blue Star Ltd.
Sabri Complex 2nd Floor
24 Residency Rd.
BANGALORE 560 025
Tel: 55660, 578881
Telex: 0845-430
Cable: BLUESTAR
A,C*,CM,E

Blue Star Ltd.
Band Box House
Prabhadevi
BOMBAY 400 025
Tel: 4933101, 4933222
Telex: 011-71051
Cable: BLUESTAR
A,M

Blue Star Ltd.
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
BOMBAY 400 025
Tel: 422-6155, 422-6556
Telex: 011-71193 BSSS IN
Cable: FROSTBLUE
A,C*,CM,E,M

Blue Star Ltd.
Kalyan, 19 Vishwas Colony
Alkapuri, **BORODA,** 390 005
Tel: 65235, 65236
Cable: BLUE STAR
A

Blue Star Ltd.
7 Hare Street
CALCUTTA 700 001
Tel: 230131, 230132
Telex: 021-7655
Cable: BLUESTAR
A,M

Blue Star Ltd.
133 Kodambakkam High Road
MADRAS 600 034
Tel: 472056, 470238
Telex: 041-379
Cable: BLUESTAR
A,M

Blue Star Ltd.
13 Community Center
New Friends Colony
NEW DELHI 110 065
Tel: 633773, 634473
Telex: 031-61120
Cable: BLUEFROST
A,C*,CM,E,M

Blue Star Ltd.
15/16 C Wellesley Rd.
PUNE 411 011
Tel: 22775
Cable: BLUE STAR
A

Blue Star Ltd.
2-2-47/1108 Bolarum Rd.
SECUNDERABAD 500 003
Tel: 72057, 72058
Telex: 0155645
Cable: BLUEFROST
A,E

Blue Star Ltd.
T.C. 7/603 Poornima
Maruthunkuzhi
TRIVANDRUM 695 013
Tel: 65799, 65820
Telex: 0884-259
Cable: BLUESTAR
E

Computer Maintenance Corporation Ltd.
115, Sarojini Devi Road
SECUNDERABAD 500 003
Tel: 310-184, 345-774
Telex: 031-2960
C**

INDONESIA

BERCA Indonesia P.T.
P.O.Box 496/Jkt.
Jl. Abdul Muis 62
JAKARTA
Tel: 21-373009
Telex: 46748 BERSAL IA
Cable: BERSAL JAKARTA
P

BERCA Indonesia P.T.
P.O.Box 2497/Jkt
Antara Bldg., 12th Floor
Jl. Medan Merdeka Selatan 17
JAKARTA-PUSAT
Tel: 21-340417, 341445
Telex: 46748 BERSAL IA
A,C,E,M

BERCA Indonesia P.T.
Jalan Kutai 24
SURABAYA
Tel: 67118
Telex: 31146 BERSAL SB
Cable: BERSAL-SURABAYA
A*,E,M,P

IRAQ

Hewlett-Packard Trading S.A.
Service Operation
Al Mansoor City 9B/3/7
BAGHDAD
Tel: 551-49-73
Telex: 212-455 HEPAIRAQ IK
C

IRELAND

Hewlett-Packard Ireland Ltd.
82/83 Lower Leeson Street
DUBLIN 2
Tel: 0001 608800
Telex: 30439
A,C,CM,E,M,P

Cardiac Services Ltd.
Kilmore Road
Artane
DUBLIN 5
Tel: (01) 351820
Telex: 30439
M

SALES & SUPPORT OFFICES
Arranged alphabetically by country

ISRAEL
Eldan Electronic Instrument Ltd.
P.O.Box 1270
JERUSALEM 91000
16, Ohaliav St.
JERUSALEM 94467
Tel: 533 221, 553 242
Telex: 25231 AB/PAKRD IL
A,M

Computation and Measurement
Systems (CMS) Ltd.
11 Masad Street
67060
TEL-AVIV
Tel: 388 388
Telex: 33569 Motil IL
C,CM,E,P

ITALY
Hewlett-Packard Italiana S.p.A
Traversa 99C
Via Giulio Petroni, 19
I-70124 **BARI**
Tel: (080) 41-07-44
C,M

Hewlett-Packard Italiana S.p.A.
Via Emilia, 51/C
I-40011 **BOLOGNA** Anzola Dell'Emilia
Tel: (051) 731061
Telex: 511630
C,E,M

Hewlett-Packard Italiana S.p.A.
Via Principe Nicola 43G/C
I-95126 **CATANIA**
Tel: (095) 37-10-87
Telex: 970291
C

Hewlett-Packard Italiana S.p.A.
Via G. Di Vittorio 9
I-20063 **CERNUSCO SUL NAVIGLIO**
(Milano)
Tel: (02) 4459041
Telex: 334632
A,C,CM,E,M,P

Hewlett-Packard Italiana S.p.A.
Via C. Colombo 49
I-20090 **TREZZANO SUL NAVIGLIO**
(Milano)
Tel: (02) 4459041
Telex: 322116
C

Hewlett-Packard Italiana S.p.A.
Via Nuova San Rocco a
Capodimonte, 62/A
I-80131 **NAPOLI**
Tel: (081) 7413544
Telex: 710698
A**,C,E,M

Hewlett-Packard Italiana S.p.A.
Viale G. Modugno 33
I-16156 **GENOVA PEGLI**
Tel: (010) 68-37-07
Telex: 215238
C,E

Hewlett-Packard Italiana S.p.A.
Via Pelizzo 15
I-35128 **PADOVA**
Tel: (049) 664888
Telex: 430315
A,C,E,M

Hewlett-Packard Italiana S.p.A.
Viale C. Pavese 340
I-00144 **ROMA EUR**
Tel: (06) 54831
Telex: 610514
A,C,E,M,P*

Hewlett-Packard Italiana S.p.A.
Via di Casellina 57/C
I-50018 **SCANDICCI-FIRENZE**
Tel: (055) 753863
C,E,M

Hewlett-Packard Italiana S.p.A.
Corso Svizzera, 185
I-10144 **TORINO**
Tel: (011) 74 4044
Telex: 221079
A*,C,E

IVORY COAST
S.I.T.E.L.
Societe Ivoirienne de
Telecommunications
Bd. Giscard d'Estaing
Carrefour Marcory
Zone 4.A.
Boite postale 2580
ABIDJAN 01
Tel: 353600
Telex: 43175
E

S.I.T.I.
Immeuble "Le General"
Av. du General de Gaulle
01 BP 161
ABIDJAN 01
Tel: 321227
C,P

JAPAN
Yokogawa-Hewlett-Packard Ltd.
152-1, Onna
ATSUGI, Kanagawa, 243
Tel: (0462) 25-0031
C,CM,E

Yokogawa-Hewlett-Packard Ltd.
Meiji-Seimei Bldg. 6F
3-1 Hon Chiba-Cho
CHIBA, 280
Tel: 472 25 7701
C,E

Yokogawa-Hewlett-Packard Ltd.
Yasuda-Seimei Hiroshima Bldg.
6-11, Hon-dori, Naka-ku
HIROSHIMA, 730
Tel: 82-241-0611

Yokogawa-Hewlett-Packard Ltd.
Towa Building
2-3, Kaigan-dori, 2 Chome Chuo-ku
KOBE, 650
Tel: (078) 392-4791
C,E

Yokogawa-Hewlett-Packard Ltd.
Kumagaya Asahi 82 Bldg
3-4 Tsukuba
KUMAGAYA, Saitama 360
Tel: (0485) 24-6563
C,CM,E

Yokogawa-Hewlett-Packard Ltd.
Asahi Shinbun Daiichi Seimei Bldg.
4-7, Hanabata-cho
KUMAMOTO, 860
Tel: (096) 354-7311
C,E

Yokogawa-Hewlett-Packard Ltd.
Shin-Kyoto Center Bldg.
614, Higashi-Shiokoji-cho
Karasuma-Nishiiru
Shiokoji-dori, Shimogyo-ku
KYOTO, 600
Tel: 075-343-0921
C,E

Yokogawa-Hewlett-Packard Ltd.
Mito Mitsui Bldg
4-73, Sanno-maru, 1 Chome
MITO, Ibaraki 310
Tel: (0292) 25-7470
C,CM,E

Yokogawa-Hewlett-Packard Ltd.
Meiji-Seimei Kokubun Bldg. 7-8
Kokubun, 1 Chome, Sendai
MIYAGI, 980
Tel: (0222) 25-1011
C,E

Yokogawa-Hewlett-Packard Ltd.
Nagoya Kokusai Center Building
47-1, Nagono, 1 Chome
Nakamura-ku
NAGOYA, 450
Tel: (052) 571-5171
C,CM,E,M

Yokogawa-Hewlett-Packard Ltd.
Saikyoren Building
1-2 Dote-machi, **OHMIYA**
Saitama 330
Tel: (0486) 45-8031

Yokogawa-Hewlett-Packard Ltd.
Chuo Bldg.,
4-20 Nishinakajima, 5 Chome
Yodogawa-ku
OSAKA, 532
Tel: (06) 304-6021
Telex: YHPOSA 523-3624
A,C,CM,E,M,P*

Yokogawa-Hewlett-Packard Ltd.
27-15, Yabe, 1 Chome
SAGAMIHARA Kanagawa, 229
Tel: 0427 59-1311

Yokogawa-Hewlett-Packard Ltd.
Daiichi Seimei Bldg.
7-1, Nishi Shinjuku, 2 Chome
Shinjuku-ku,**TOKYO** 160
Tel: 03-348-4611
C,E

Yokogawa-Hewlett-Packard Ltd.
29-21 Takaido-Higashi, 3 Chome
Suginami-ku **TOKYO** 168
Tel: (03) 331-6111
Telex: 232-2024 YHPTOK
A,C,CM,E,M,P*

Yokogawa Hokushin Electric Corp.
9-32 Nokacho 2 Chome
2 Chome Musashino-shi
TOKYO, 180
Tel: (0422) 54-1111
Telex: 02822-421 YEW MTK J
A

Yokogawa-Hewlett-Packard Ltd.
Meiji-Seimei
Utsunomiya Odori Building
1-5 Odori, 2 Chome
UTSUNOMIYA, Tochigi 320
Tel: (0286) 33-1153
C,E

Yokogawa-Hewlett-Packard Ltd.
Yasuda Seimei Yokohama Nishiguchi Bldg.
30-4 Tsuruya-cho, 3 Chome
YOKOHAMA 221
Tel: (045) 312-1252
C,E

JORDAN
Scientific and Medical Supplies Co.
P.O. Box 1387
AMMAN
Tel: 24907, 39907
Telex: 21456 SABCO JO
C,E,M,P

KENYA
ADCOM Ltd., Inc., Kenya
P.O.Box 30070
NAIROBI
Tel: 331955
Telex: 22639
E,M

KOREA
Samsung Hewlett-Packard Co. Ltd.
Dongbang Yeoeuido Building
12-16th Floors
36-1 Yeoeuido-dong
Yongdeungpo-ku
SEOUL
Tel: 784-2666, 784-4666
Telex: 25166 SAMSAN K
A,C,CM,E,M,P

Young In Scientific Co., Ltd.
Youngwha Building
547 Shinsa Dong, Kangnam-ku
SEOUL 135
Tel: 5467771
Telex: K23457 GINSCO
A

KUWAIT
Al-Khaldiya Trading & Contracting
P.O. Box 830
SAFAT
Tel: 424910, 411726
Telex: 22481 AREEG KT
Cable: VISCOUNT
E,M,A

Gulf Computing Systems
P.O. Box 25125
SAFAT
Tel: 435969
Telex: 23648
P

Photo & Cine Equipment
P.O. Box 270
SAFAT
Tel: 2445111
Telex: 22247 MATIN KT
Cable: MATIN KUWAIT
P

W.J. Towell Computer Services
P.O. Box 5897
SAFAT
Tel: 2462640
Telex: 30336 TOWELL KT
C

LEBANON
Computer Information Systems S.A.L.
Chammas Building
P.O. Box 11-6274 Dora
BEIRUT
Tel: 89 40 73
Telex: 42309
C,E,M,P

LIBERIA
Unichemicals Inc.
P.O. Box 4509
MONROVIA
Tel: 224282
Telex: 4509
E

MADAGASCAR
Technique et Precision
12, rue de Nice
P.O. Box 1227
101 **ANTANANARIVO**
Tel: 22090
Telex: 22255
P

LUXEMBOURG
Hewlett-Packard Belgium S.A./N.V.
Blvd de la Woluwe, 100
Woluwedal
B-1200 **BRUSSELS**
Tel: (02) 762-32-00
Telex: 23-494 paloben bru
A,C,CM,E,M,P

MALAYSIA
Hewlett-Packard Sales (Malaysia)
Sdn. Bhd.
9th Floor
Chung Khiaw Bank Building
46, Jalan Raja Laut
KUALA LUMPUR
Tel: 03-986555
Telex: 31011 HPSM MA
A,C,E,M,P*

Protel Engineering
P.O.Box 1917
Lot 6624, Section 64
23/4 Pending Road
Kuching, **SARAWAK**
Tel: 36299
Telex: 70904 PROMAL MA
Cable: PROTELENG
A,E,M

MALTA
Philip Toledo Ltd.
Birkirkara P.O. Box 11
Notabile Rd.
MRIEHEL
Tel: 447 47, 455 66
Telex: 1649
E,M,P

MAURITIUS
Blanche Birger Co. Ltd.
18, Jules Koenig Street
PORT LOUIS
Tel: 20828
Telex: 4296
P

MEXICO
Hewlett-Packard de Mexico, S.A.
Francisco J. Allan #30
Colonia Nueva
Los Angeles 27140
COAHUILA, Torreon
Tel: 37220
P

Hewlett-Packard de Mexico, S.A.
Monti Morelos 299
Fraccionamiento Loma Bonita 45060
GUADALAJARA, Jalisco
Tel: 316630/314600
Telex: 0684 186 ECOME
P

MEXICO (Cont'd)

Microcomputadoras Hewlett-Packard, S.A.
Monti Pelvoux 115
LOS LOMAS, Mexico, D.F.
Tel: 520-9127
P

Hewlett-Packard Mexicana, S.A. de C.V.
Av. Periferico Sur No. 6501
Tepepan, Xochimilco
16020 **MEXICO D.F.**
Tel: 6-76-46-00
Telex: 17-74-507 HEWPACK MEX
A,C,CM,E,M,P

Hewlett-Packard De Mexico (Polanco)
Avenida Ejercito Nacional #579
2day3^{er} piso
Colonia Granada 11560
MEXICO D.F.
Tel: 254-4433
P

Hewlett-Packard De Mexico, S.A. de C.V.
Czda. del Valle
409 Ote. 4th Piso
Colonia del Valle
Municipio de Garza Garciá
66220 **MONTERREY,** Nuevo León
Tel: 78 42 41
Telex: 038 410
P

MOROCCO

Etablissement Hubert Dolbeau & Fils
81 rue Karatchi
B.P. 11133
CASABLANCA
Tel: 3041-82, 3068-38
Telex: 23051, 22822
E

Gerep
2, rue Agadir
Boite Postale 156
CASABLANCA 01
Tel: 272093, 272095
Telex: 23 739
P

Sema-Maroc
Dept. Seric
6, rue Lapebie
CASABLANCA
Tel: 260980
Telex: 21641
C,P

NETHERLANDS

Hewlett-Packard Nederland B.V.
Startbaan 16
1187 XR **AMSTELVEEN**
P.O. Box 667
NL1180 AR **AMSTELVEEN**
Tel: (020) 547-6911
Telex: 13 216 HEPA NL
A,C,CM,E,M,P

Hewlett-Packard Nederland B.V.
Bongerd 2
NL 2906VK **CAPELLE A/D IJSSEL**
P.O. Box 41
NL 2900AA **CAPELLE A/D IJSSEL**
Tel: (10) 51-64-44
Telex: 21261 HEPAC NL
C,E

Hewlett-Packard Nederland B.V.
Pastoor Petersstraat 134-136
NL 5612 LV **EINDHOVEN**
P.O. Box 2342
NL 5600 CH **EINDHOVEN**
Tel: (040) 326911
Telex: 51484 hepae nl
A,C,E,M,P

NEW ZEALAND

Hewlett-Packard (N.Z.) Ltd.
5 Owens Road
P.O. Box 26-189
Epsom, **AUCKLAND**
Tel: 687-159
Cable: HEWPAK Auckland
C,CM,E,P*

Hewlett-Packard (N.Z.) Ltd.
4-12 Cruickshank Street
Kilbirnie, **WELLINGTON 3**
P.O. Box 9443
Courtenay Place, **WELLINGTON 3**
Tel: 877-199
Cable: HEWPACK Wellington
C,CM,E,P

Northrop Instruments & Systems Ltd.
369 Khyber Pass Road
P.O. Box 8602
AUCKLAND
Tel: 794-091
Telex: 60605
A,M

Northrop Instruments & Systems Ltd.
110 Mandeville St.
P.O. Box 8388
CHRISTCHURCH
Tel: 488-873
Telex: 4203
A,M

Northrop Instruments & Systems Ltd.
Sturdee House
85-87 Ghuznee Street
P.O. Box 2406
WELLINGTON
Tel: 850-091
Telex: NZ 3380
A,M

NIGERIA

Elmeco Nigeria Ltd.
46, Calcutta Crescent Apapa
P.O. Box 244and
LAGOS
E

NORTHERN IRELAND

See United Kingdom

NORWAY

Hewlett-Packard Norge A/S
Folke Bernadottes vei 50
P.O. Box 3558
N-5033 **FYLLINGSDALEN** (Bergen)
Tel: 0047/5/16 55 40
Telex: 76621 hpnas n
C,E,M

Hewlett-Packard Norge A/S
Osterndalen 16-18
P.O. Box 34
N-1345 **ÖSTERÅS**
Tel: 0047/2/17 11 80
Telex: 76621 hpnas n
A,C,CM,E,M,P

OMAN

Khimjil Ramdas
P.O. Box 19
MUSCAT/SULTANATE OF OMAN
Tel: 745601
Telex: 5289 BROKER MB MUSCAT
P

Suhail & Saud Bahwan
P.O.Box 169
MUSCAT/SULTANATE OF OMAN
Tel: 734201
Telex: 5274 BAHWAN MB
E

Imtac LLC
P.O. Box 8676
MUTRAH/SULTANATE OF OMAN
Tel: 601695
Telex: 5741 Tawoos On
A,C,M

PAKISTAN

Mushko & Company Ltd.
House No. 16, Street No. 16
Sector F-6/3
ISLAMABAD
Tel: 824545
Cable: FEMUS Islamabad
A,E,M,P*

Mushko & Company Ltd.
Oosman Chambers
Abdullah Haroon Road
KARACHI 0302
Tel: 524131, 524132
Telex: 2894 MUSKO PK
Cable: COOPERATOR Karachi
A,E,M,P*

PANAMA

Electronico Balboa, S.A.
Calle Samuel Lewis, Ed. Alfa
Apartado 4929
PANAMA 5
Tel: 64-2700
Telex: 3483 ELECTRON PG
A,CM,E,M,P

PERU

Cía Electro Médica S.A.
Los Flamencos 145, Ofc. 301/2
San Isidro
Casilla 1030
LIMA 1
Tel: 41-4325, 41-3705
Telex: Pub. Booth 25306 PEC PISIDR
CM,E,M,P

SAMS
Arenida Republica de Panama 3534
San Isidro, LIMA
Tel: 419928/417108
Telex: 20450 PE LIBERTAD
A,C,P

PHILIPPINES

The Online Advanced Systems Corp.
2nd Floor, Electra House
115-117 Esteban Street
Legaspi Village, Makati
P.O. Box 1510
Metro **MANILA**
Tel: 815-38-10 (up to 16)
Telex: 63274 ONLINE PN
A,C,E,M,P

PORTUGAL

Mundinter Intercambio
Mundial de Comércio S.A.R.L.
Av. Antonio Augusto Aguiar 138
Apartado 2761
LISBON
Tel: (19) 53-21-31, 53-21-37
Telex: 16691 munter p
M

Soquimica
Av. da Liberdade, 220-2
1298 **LISBOA** Codex
Tel: 56-21-82
Telex: 13316 SABASA
A

Telectra-Empresa Técnica de Equipmentos Eléctricos S.A.R.L.
Rua Rodrigo da Fonseca 103
P.O. Box 2531
LISBON 1
Tel: (19) 68-60-72
Telex: 12598
CM,E

C.P.C.S.I.
Rua de Costa Cabral 575
4200 **PORTO**
Tel: 499174/495173
Telex: 26054
C,P

PUERTO RICO

Hewlett-Packard Puerto Rico
101 Muñoz Rivera Av
Esu. Calle Ochoa
HATO REY, Puerto Rico 00918
Tel: (809) 754-7800
A,C,CM,M,E,P

QATAR

Computer Arabia
P.O. Box 2750
DOHA
Tel: 428555
Telex: 4806 CHPARB
P

Nasser Trading & Contracting
P.O.Box 1563
DOHA
Tel: 422170
Telex: 4439 NASSER DH
M

SAUDI ARABIA

Modern Electronics Establishment
Hewlett-Packard Division
P.O. Box 281
Thouqbah
AL-KHOBAR 31952
Tel: 895-1760, 895-1764
Telex: 671 106 HPMEEK SJ
Cable: ELECTA AL-KHOBAR
C,E,M

Modern Electronics Establishment
Hewlett-Packard Division
P.O. Box 1228
JEDDAH
Tel: 644 96 28
Telex: 4027 12 FARNAS SJ
Cable: ELECTA JEDDAH
A,C,CM,E,M,P

Modern Electronics Establishment
Hewlett-Packard Division
P.O.Box 22015
RIYADH 11495
Tel: 476-3030
Telex: 202049 MEERYD SJ
A,C,CM,E,M,P

Abdul Ghani El Ajou Corp.
P.O. Box 78
RIYADH
Tel: 40 41 717
Telex: 200 931 EL AJOU
P

SCOTLAND

See United Kingdom

SENEGAL

Societe Hussein Ayad & Cie.
76, Avenue Georges Pompidou
B.P. 305
DAKAR
Tel: 32339
Cable: AYAD-Dakar
E

Moneger Distribution S.A.
1, Rue Parent
B.P. 148
DAKAR
Tel: 215 671
Telex: 587
P

Systeme Service Conseil (SSC)
14, Avenue du Parachois
DAKAR ETOILE
Tel: 219976
Telex: 577
C,P

SINGAPORE

Hewlett-Packard Singapore (Sales) Pte. Ltd.
08-00 Inchcape House
450-2 Alexandra Road
Alexandra P.O. Box 58
SINGAPORE, 9115
Tel: 4731788
Telex: 34209 HPSGSO RS
Cable: HEWPACK, Singapore
A,C,E,M,P

Dynamar International Ltd.
Unit 05-11 Block 6
Kolam Ayer Industrial Estate
SINGAPORE 1334
Tel: 747-6188
Telex: 26283 RS
CM

SOUTH AFRICA

Hewlett-Packard So Africa (Pty.) Ltd.
P.O. Box 120
Howard Place **CAPE PROVINCE** 7450
Pine Park Center, Forest Drive, Pinelands
CAPE PROVINCE 7405
Tel: (021) 53 7954
Telex: 57-20006
A,C,CM,E,M,P

Hewlett-Packard So Africa (Pty.) Ltd.
2nd Floor Juniper House
92 Overport Drive
DURBAN 4067
Tel: (031) 28-4178
Telex: 6-22954
C

Hewlett-Packard So Africa (Pty.) Ltd.
6 Linton Arcade
511 Cape Road
Linton Grange
PORT ELIZABETH 6001
Tel: 041-301201
Telex: 24-2916
C

SALES & SUPPORT OFFICES
Arranged alphabetically by country

SOUTH AFRICA (Cont'd)
Hewlett-Packard So Africa (Pty.) Ltd.
Fountain Center
Kalkden Str.
Monument Park Ext 2
PRETORIA 0105
Tel: (012) 45 57258
Telex: 3-21063
C,E

Hewlett-Packard So Africa (Pty.) Ltd.
Private Bag Wendywood
SANDTON 2144
Tel: 802-5111, 802-5125
Telex: 4-20877 SA
Cable: HEWPACK Johannesburg
A,C,CM,E,M,P

SPAIN
Hewlett-Packard Española S.A.
Calle Entenza, 321
08029 **BARCELONA**
Tel: 3/322 24 51, 321 73 54
Telex: 52603 hpbee
A,C,E,M,P

Hewlett-Packard Española S.A.
Calle San Vicente S/N
Edificio Albia II-7B
48001 **BILBAO**
Tel: 4/423 83 06
A,C,E,M

Hewlett-Packard Española S.A.
Crta. de la Coruña, Km. 16, 400
Las Rozas
E-**MADRID**
Tel: (1) 637.00.11
Telex: 23515 HPE
C,M

Hewlett-Packard Española S.A.
Avda. S. Francisco Javier, S/N
Planta 10. Edificio Sevilla 2
41005 **SEVILLA**
Tel: 54/64 44 54
Telex: 72933
A,C,M,P

Hewlett-Packard Española S.A.
Isabel La Catolica, 8
46004 **VALENCIA**
Tel: 0034/6/351 59 44
C,P

SWEDEN
Hewlett-Packard Sverige AB
Ostra Tullgatan 3
S-21128 **MALMÖ**
Tel: (040) 70270
Telex: (854) 17886 (via Spånga office)
C,P

Hewlett-Packard Sverige AB
Skalholtsgatan 9, Kista
Box 19
S-16393 **SPÅNGA**
Tel: (08) 750-2000
Telex: (854) 17886
Telefax: (08) 7527781
A,C,CM,E,M,P

Hewlett-Packard Sverige AB
Frötallsgatan 30
S-42132 **VÄSTRA-FRÖLUNDA** (Gothenburg)
Tel: (031) 49-09-50
Telex: (854) 17886 (via Spånga office)
A,C,CM,E,M,P

SUDAN
Mediterranean Engineering & Trading Co. Ltd.
P.O. Box 1025
KHARTOUM
Tel: 41184
Telex: 24052
C,P

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Clarastrasse 12
CH-4058 **BASEL**
Tel: (61) 33-59-20
A

Hewlett-Packard (Schweiz) AG
7, rue du Bois-du-Lan
Case postale 365
CH-1217 **MEYRIN** 1
Tel: (0041) 22-83-11-11
Telex:27333 HPAG CH
C,CM

Hewlett-Packard (Schweiz) AG
Allmend 2
CH-8967 **WIDEN**
Tel: (0041) 57 31 21 11
Telex: 53933 hpag ch
Cable: HPAG CH
A,C,CM,E,M,P

SYRIA
General Electronic Inc.
Nuri Basha Ahnaf Ebn Kays Street
P.O. Box 5781
DAMASCUS
Tel: 33-24-87
Telex: 411 215
Cable: ELECTROBOR DAMASCUS
E

Middle East Electronics
P.O.Box 2308
Abu Rumaneh
DAMASCUS
Tel: 33 45 92
Telex: 411 771
M

TAIWAN
Hewlett-Packard Taiwan
Kaohsiung Office
11/F, 456, Chung Hsiao 1st Road
KAOHSIUNG
Tel: (07) 2412318
C,E

Hewlett-Packard Taiwan
8th Floor, Hewlett-Packard Building
337 Fu Hsing North Road
TAIPEI
Tel: (02) 712-0404
Telex: 24439 HEWPACK
Cable:HEWPACK Taipei
A,C,CM,E,M,P

Ing Lih Trading Co.
3rd Floor, 7 Jen-Ai Road, Sec. 2
TAIPEI 100
Tel: (02) 3948191
Cable: INGLIH Taipei
A

THAILAND
Unimesa Co. Ltd.
30 Patpong Ave., Suriwong
BANGKOK 5
Tel: 235-5727
Telex: 84439 Simonco TH
Cable: UNIMESA Bangkok
A,C,E,M

Bangkok Business Equipment Ltd.
5/5-6 Dejo Road
BANGKOK
Tel: 234-8670, 234-8671
Telex: 87699-BEQUIPT TH
Cable: BUSIQUIPT Bangkok
P

TOGO
Societe Africaine De Promotion
Immeuble Sagap
22, Rue d'Atakpame
B.P. 4150
LOME
Tel: 21-62-88
Telex: 5304
P

TRINIDAD & TOBAGO
Caribbean Telecoms Ltd.
Corner McAllister Street &
Eastern Main Road, Laventille
P.O. Box 732
PORT-OF-SPAIN
Tel: 624-4213
Telex: 22561 CARTEL WG
Cable: CARTEL, PORT OF SPAIN
CM,E,M,P

Computer and Controls Ltd.
P.O. Box 51
66 Independence Square
PORT-OF-SPAIN
Tel: 62-279-85
Telex: 3000 POSTLX WG, ACCT LOO9O AGENCY 1264
A,P

Feral Assoc.
8 Fitzgerald Lane
PORT-OF-SPAIN
Tel: 62-36864, 62-39255
Telex: 22432 FERALCO
Cable: FERALCO
M

TUNISIA
Precision Electronique S.A.R.L.
31 Avenue de la Liberte
TUNIS
Tel: 893937
Telex: 13238
P

Tunisie Electronique S.A.R.L.
94, Av. Jugurtha, Mutuelleville
1002 **TUNIS-BELVEDERE**
Tel: 280144
Telex: 13238
C,E,P

Corema S.A.
23, bis Rue de Marseille
TUNIS
Tel: 253-821
Telex: 14812 CABAM TN
M

TURKEY
E.M.A
Mediha Eldem Sokak No. 41/6
Yenisehir
ANKARA
Tel: 319175
Telex: 46912 KTX TR
Cable: EMATRADE ANKARA
M

Teknim Company Ltd.
Iran Caddesi No. 7
Kavaklidere
ANKARA
Tel: 275800
Telex: 42155 TKNM TR
E,CM

Saniva Bilgisayar Sistemleri A.S.
Buyukdere Caddesi 103/6
Gayrettene
ISTANBUL
Tel: 1727030
Telex: 26345 SANI TR
C,P

Best Inc.
Esentepe, Gazeteciler Sitesi
Keskin Kalemy
Sokak 6/3, Gayrettepe
ISTANBUL
Tel: 1721328
Telex: 42490
A

UNITED ARAB EMIRATES
Emitac Ltd.
P.O. Box 1641
SHARJAH
Tel: 591181
Telex: 68136 EMITAC EM
Cable: EMITAC SHARJAH
E,C,M,P,A

Emitac Ltd.
P.O. Box 2711
ABU DHABI
Tel: 820419-20
Cable: EMITACH ABUDHABI

Emitac Ltd.
P.O. Box 8391
DUBAI,
Tel: 377591

Emitac Ltd.
P.O. Box 473
RAS AL KHAIMAH
Tel: 28133, 21270

UNITED KINGDOM

GREAT BRITAIN
Hewlett-Packard Ltd.
Trafalgar House
Navigation Road
ALTRINCHAM
Cheshire WA14 1NU
Tel: 061 928 6422
Telex: 668068
A,C,E,M,P

Hewlett-Packard Ltd.
Miller House
The Ring, **BRACKNELL**
Berks RG12 1XN
Tel: 0344 424898
Telex: 848733
E

Hewlett-Packard Ltd.
Elstree House, Elstree Way
BOREHAMWOOD, Herts WD6 1SG
Tel: 01 207 5000
Telex: 8952716
C,E

Hewlett-Packard Ltd.
Oakfield House, Oakfield Grove
Clifton **BRISTOL,** Avon BS8 2BN
Tel: 0272 736806
Telex: 444302
C,E,P

Hewlett-Packard Ltd.
Bridewell House
9 Bridewell Place
LONDON EC4V 6BS
Tel: 01 583 6565
Telex: 298163
C,P

Hewlett-Packard Ltd.
Pontefract Road
NORMANTON, West Yorkshire WF6 1RN
Tel: 0924 895566
Telex: 557355
C,P

Hewlett-Packard Ltd.
The Quadrangle
106-118 Station Road
REDHILL, Surrey RH1 1PS
Tel: 0737 68655
Telex: 947234
C,E,P

Hewlett-Packard Ltd.
Avon House
435 Stratford Road
Shirley, **SOLIHULL,** West Midlands
B90 4BL
Tel: 021 745 8800
Telex: 339105
C,E,P

Hewlett-Packard Ltd.
West End House
41 High Street, West End
SOUTHAMPTON
Hampshire S03 3DQ
Tel: 0703 476767
Telex: 477138
C,P

Hewlett-Packard Ltd.
Harmon House
No. 1 George Street
UXBRIDGE, Middlesex UX8 1YH
Tel: 895 720 20
Telex: 893134/5
C,CM,E,M,P

Hewlett-Packard Ltd.
King Street Lane
Winnersh, **WOKINGHAM**
Berkshire RG11 5AR
Tel: 0734 784774
Telex: 847178
A,C,E,M,P

IRELAND

NORTHERN IRELAND
Hewlett-Packard (Ireland) Ltd.
Carrickfergus Industrial Centre
75 Belfast Road, Carrickfergus
BELFAST BT38 8PH
Tel: 09603 67333
Telex: 747626
C,E

SCOTLAND
Hewlett-Packard Ltd.
8 Woodside Place
GLASGOW, G3 7QF
Tel: 041 332 6232
Telex: 779615
C,E

Hewlett-Packard Ltd.
SOUTH QUEENSFERRY
West Lothian, EH30 9TG
Tel: 031 331 1188
Telex: 72682
C,CM,E,M,P

UNITED STATES

Alabama
Hewlett-Packard Co.
700 Century Park South, Suite 128
BIRMINGHAM, AL 35226
Tel: (205) 822-6802
A,C,M,P*

Hewlett-Packard Co.
420 Wynn Drive
HUNTSVILLE, AL 35805
Tel: (205) 830-2000
C,CM,E,M*

Alaska
Hewlett-Packard Co.
3601 C St., Suite 1416
ANCHORAGE, AK 99503
Tel: (907) 563-8855
C,E

Arizona
Hewlett-Packard Co.
8080 Pointe Parkway West
PHOENIX, AZ 85044
Tel: (602) 273-8000
A,C,CM,E,M,P

Hewlett-Packard Co.
3400 East Britannia Dr.
Bldg. C, Suite 124
TUCSON, AZ 85706
Tel: (602) 573-7400
C,E,M**

California
Hewlett-Packard Co.
99 South Hill Dr.
BRISBANE, CA 94005
Tel: (415) 330-2500
C

Hewlett-Packard Co.
5060 E. Clinton Avenue, Suite 102
FRESNO, CA 93727
Tel: (209) 252-9652
C,M

Hewlett-Packard Co.
1421 S. Manhattan Av.
FULLERTON, CA 92631
Tel: (714) 999-6700
C,CM,E,M

Hewlett-Packard Co.
7408 Hollister Ave. #A
GOLETA, CA 93117
Tel: (805) 685-6100
C,E

Hewlett-Packard Co.
5400 W. Rosecrans Blvd.
LAWNDALE, CA 90260
Tel: (213) 643-7500
Telex: 910-325-6608
C,M

Hewlett-Packard Co.
2525 Grand Avenue
Long Beach, CA 90815
Tel: (213) 498-1111
C

Hewlett-Packard Co.
3155 Porter Drive
PALO ALTO, CA 94304
Tel: (415) 857-8000
C,E

Hewlett-Packard Co.
4244 So. Market Court, Suite A
SACRAMENTO, CA 95834
Tel: (916) 929-7222
A*,C,E,M

Hewlett-Packard Co.
9606 Aero Drive
SAN DIEGO, CA 92123
Tel: (619) 279-3200
C,CM,E,M

Hewlett-Packard Co.
5725 W. Las Positas Blvd.
Pleasanton, CA 94566
Tel: (415) 460-0282
C

Hewlett-Packard Co.
3003 Scott Boulevard
SANTA CLARA, CA 95054
Tel: (408) 988-7000
Telex: 910-338-0586
A,C,CM,E

Hewlett-Packard Co.
2150 W. Hillcrest Dr.
THOUSAND OAKS, CA 91320
(805) 373-7000
C,CM,E

Colorado
Hewlett-Packard Co.
2945 Center Green Court South
Suite A
BOULDER, CO 80301
Tel: (303) 938-3005
A,C,E

Hewlett-Packard Co.
24 Inverness Place, East
ENGLEWOOD, CO 80112
Tel: (303) 649-5000
A,C,CM,E,M

Connecticut
Hewlett-Packard Co.
500 Sylvan Av.
BRIDGEPORT, CT 06606
Tel: (203) 371-6454
C,E

Hewlett-Packard Co.
47 Barnes Industrial Road South
WALLINGFORD, CT 06492
Tel: (203) 265-7801
A,C,CM,E,M

Florida
Hewlett-Packard Co.
2901 N.W. 62nd Street
FORT LAUDERDALE, FL 33309
Tel: (305) 973-2600
C,E,M,P*

Hewlett-Packard Co.
6800 South Point Parkway
Suite 301
JACKSONVILLE, FL 32216
Tel: (904) 398-0663
C*,M**

Hewlett-Packard Co.
6177 Lake Ellenor Drive
ORLANDO, FL 32809
Tel: (305) 859-2900
A,C,CM,E,P*

Hewlett-Packard Co.
4700 Bayou Blvd.
Building 5
PENSACOLA, FL 32503
Tel: (904) 476-8422
A,C,M

Hewlett-Packard Co.
5550 W. Idlewild, 150
TAMPA, FL 33614
Tel: (813) 884-3282
C,E,M,P

Georgia
Hewlett-Packard Co.
2000 South Park Place
ATLANTA, GA 30339
Tel: (404) 955-1500
Telex: 810-766-4890
A,C,CM,E,M,P*

Hewlett-Packard Co.
3607 Parkway Lane
Suite 300
NORCROSS, GA 30092
Tel: (404) 448-1894
C,E,P

Hawaii
Hewlett-Packard Co.
Kawaiahao Plaza, Suite 190
567 South King Street
HONOLULU, HI 96813
Tel: (808) 526-1555
A,C,E,M

Idaho
Hewlett-Packard Co.
11309 Chinden Blvd.
BOISE, ID 83707
Tel: (208) 323-2700
C

Illinois
Hewlett-Packard Co.
304 Eldorado Road
P.O. Box 1607
BLOOMINGTON, IL 61701
Tel: (309) 662-9411
C,M**

Hewlett-Packard Co.
525 W. Monroe, 1308
CHICAGO, IL 60606
Tel: (312) 930-0010
C

Hewlett-Packard Co.
1200 East Diehl Road
NAPERVILLE, IL 60566
Tel: (312) 357-8800
C

Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800
Telex: 910-687-1066
A,C,CM,E,M

Indiana
Hewlett-Packard Co.
11911 N. Meridian St.
CARMEL, IN 46032
Tel: (317) 844-4100
A,C,CM,E,M

Hewlett-Packard Co.
3702 Rupp Drive
FT. WAYNE, IN 46815
Tel: (219) 482-4283
C,E

Iowa
Hewlett-Packard Co.
4070 22nd Av. SW
CEDAR RAPIDS, IA 52404
Tel: (319) 390-4250
C,E,M

Hewlett-Packard Co.
4201 Corporate Dr.
WEST DES MOINES, IA 50265
Tel: (515) 224-1435
A**,C,M**

Kansas
Hewlett-Packard Co.
7804 East Funston Road, 203
WICHITA, KS 67207
Tel: (316) 684-8491
C,E

Kentucky
Hewlett-Packard Co.
10300 Linn Station Road, 100
LOUISVILLE, KY 40223
Tel: (502) 426-0100
A,C,M

Louisiana
Hewlett-Packard Co.
160 James Drive East
ST. ROSE, LA 70087
P.O. Box 1449
KENNER, LA 70063
Tel: (504) 467-4100
A,C,E,M,P

Maryland
Hewlett-Packard Co.
3701 Koppers Street
BALTIMORE, MD 21227
Tel: (301) 644-5800
Telex: 710-862-1943
A,C,CM,E,M

Hewlett-Packard Co.
2 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 948-6370
A,C,CM,E,M

Massachusetts
Hewlett-Packard Co.
1775 Minuteman Road
ANDOVER, MA 01810
Tel: (617) 682-1500
A,C,CM,E,M,P*

Hewlett-Packard Co.
32 Hartwell Avenue
LEXINGTON, MA 02173
Tel: (617) 861-8960
C,E

Michigan
Hewlett-Packard Co.
4326 Cascade Road S.E.
GRAND RAPIDS, MI 49506
Tel: (616) 957-1970
C,M

Hewlett-Packard Co.
39550 Orchard Hill Place Drive
NOVI, MI 48020
Tel: (313) 349-9200
A,C,E,M

Hewlett-Packard Co.
1771 W. Big Beaver Road
TROY, MI 48084
Tel: (313) 643-6474
C

Minnesota
Hewlett-Packard Co.
2025 W. Larpenteur Ave.
ST. PAUL, MN 55113
Tel: (612) 644-1100
A,C,CM,E,M

Missouri
Hewlett-Packard Co.
1001 E. 101st Terrace Suite 120
KANSAS CITY, MO 64131-3368
Tel: (816) 941-0411
A,C,CM,E,M

Hewlett-Packard Co.
13001 Hollenberg Drive
BRIDGETON, MO 63044
Tel: (314) 344-5100
A,C,E,M

Nebraska
Hewlett-Packard
10824 Old Mill Rd., Suite 3
OMAHA, NE 68154
Tel: (402) 334-1813
C,E,M

New Jersey
Hewlett-Packard Co.
120 W. Century Road
PARAMUS, NJ 07653
Tel: (201) 265-5000
A,C,CM,E,M

Hewlett-Packard Co.
20 New England Av. West
PISCATAWAY, NJ 08854
Tel: (201) 562-6100
A,C,CM,E

New Mexico
Hewlett-Packard Co.
7801 Jefferson N.E.
ALBUQUERQUE, NM 87109
Tel: (505) 292-1330
C,E,M

New York
Hewlett-Packard Co.
5 Computer Drive South
ALBANY, NY 12205
Tel: (518) 458-1550
A,C,E,M

Hewlett-Packard Co.
9600 Main Street
CLARENCE, NY 14031
Tel: (716) 759-8621
C,E

Hewlett-Packard Co.
200 Cross Keys Office Park
FAIRPORT, NY 14450
Tel: (716) 223-9950
A,C,CM,E,M

Hewlett-Packard Co.
7641 Henry Clay Blvd.
LIVERPOOL, NY 13088
Tel: (315) 451-1820
A,C,CM,E,M

Hewlett-Packard Co.
No. 1 Pennsylvania Plaza
55th Floor
34th Street & 8th Avenue
MANHATTAN NY 10119
Tel: (212) 971-0800
C,M*

Hewlett-Packard Co.
15 Myers Corner Rd.
Hollowbrook Park, Suite 20
WAPPINGER FALLS, NY 12590
CM,E

Hewlett-Packard Co.
250 Westchester Avenue
WHITE PLAINS, NY 10604
Tel: (914) 684-6100
C,CM,E

Hewlett-Packard Co.
3 Crossways Park West
WOODBURY, NY 11797
Tel: (516) 682-7800
A,C,CM,E,M

SALES & SUPPORT OFFICES
Arranged alphabetically by country

UNITED STATES (Cont'd)

North Carolina
Hewlett-Packard Co.
305 Gregson Dr.
CARY, NC 27511
Tel: (919) 467-6600
C,CM,E,M,P*

Hewlett-Packard Co.
9600-H Southern Pine Blvd.
CHARLOTTE, NC 28210
Tel: (704) 527-8780
C*

Hewlett-Packard Co.
5605 Roanne Way
GREENSBORO, NC 27420
Tel: (919) 852-1800
A,C,CM,E,M,P*

Ohio
Hewlett-Packard Co.
2717 S. Arlington Road
AKRON, OH 44312
Tel: (216) 644-2270
C,E

Hewlett-Packard Co.
23200 Chagrin Blvd #100
BEACHWOOD, OH 44122
Tel: (216) 292-4677
C,P

Hewlett-Packard Co.
9920 Carver Road
CINCINNATI, OH 45242
Tel: (513) 891-9870
C,M

Hewlett-Packard Co.
16500 Sprague Road
CLEVELAND, OH 44130
Tel: (216) 243-7300
A,C,CM,E,M

Hewlett-Packard Co.
9080 Springboro Pike
MIAMISBURG, OH 45342
Tel: (513) 433-2223
A,C,CM,E*,M

Hewlett-Packard Co.
One Maritime Plaza, 5th Floor
720 Water Street
TOLEDO, OH 43604
Tel: (419) 242-2200
C

Hewlett-Packard Co.
675 Brooksedge Blvd.
WESTERVILLE, OH 43081
Tel: (614) 891-3344
C,CM,E*

Oklahoma
Hewlett-Packard Co.
3525 N.W. 56th St.
Suite C-100
OKLAHOMA CITY, OK 73112
Tel: (405) 946-9499
C,E*,M

Hewlett-Packard Co.
3840 S. 103rd E. Ave., 100
TULSA, OK 74146
Tel: (918) 665-3300
A**,C,E,M*,P*

Oregon
Hewlett-Packard Co.
9255 S. W. Pioneer Court
WILSONVILLE, OR 97070
Tel: (503) 682-8000
A,C,E*,M

Pennsylvania
Hewlett-Packard Co.
50 Dorchester Rd.
HARRISBURG, PA 17112
Tel: (717) 657-5900
C

Hewlett-Packard Co.
111 Zeta Drive
PITTSBURGH, PA 15238
Tel: (412) 782-0400
A,C,E,M

Hewlett-Packard Co.
2750 Monroe Boulevard
VALLEY FORGE, PA 19482
Tel: (215) 666-9000
A,C,CM,E,M

South Carolina
Hewlett-Packard Co.
Brookside Park, Suite 122
1 Harbison Way
COLUMBIA, SC 29210
Tel: (803) 732-0400
C,M

Hewlett-Packard Co.
555 N. Pleasantburg Dr.
Suite 107
GREENVILLE, SC 29607
Tel: (803) 232-8002
C

Tennessee
Hewlett-Packard Co.
One Energy Centr. 200
Pellissippi Pkwy.
KNOXVILLE, TN 37932
Tel: (615) 966-4747
A,C,M

Hewlett-Packard Co.
3070 Directors Row
Directors Square
MEMPHIS, TN 38131
Tel: (901) 346-8370
A,C,M

Hewlett-Packard Co.
220 Great Circle Road, Suite 116
NASHVILLE, TN 37228
Tel: (615) 255-1271
C,M,P*

Texas
Hewlett-Packard Co.
1826-P Kramer Lane
AUSTIN, TX 78758
Tel: (512) 835-6771
C,E,P*

Hewlett-Packard Co.
5700 Cromo Dr
EL PASO, TX 79912
Tel: (915) 833-4400
C,E*,M**

Hewlett-Packard Co.
3952 Sandshell Drive
FORT WORTH, TX 76137
Tel: (817) 232-9500
C

Hewlett-Packard Co.
10535 Harwin Drive
HOUSTON, TX 77036
Tel: (713) 776-6400
A,C,E,M,P*

Hewlett-Packard Co.
511 E. John W. Carpenter Fwy.
Royal Tech. Center 100
IRVING, TX 75062
Tel: (214) 556-1950
C,E

Hewlett-Packard Co.
109 E. Toronto, Suite 100
McALLEN, TX 78503
Tel: (512) 630-3030
C

Hewlett-Packard Co.
930 E. Campbell Rd.
RICHARDSON, TX 75081
Tel: (214) 231-6101
A,C,CM,E,M,P*

Hewlett-Packard Co.
1020 Central Parkway South
SAN ANTONIO, TX 78216
Tel: (512) 494-9336
A,C,E,M,P*

Utah
Hewlett-Packard Co.
3530 W. 2100 South
SALT LAKE CITY, UT 84119
Tel: (801) 974-1700
A,C,E,M

Virginia
Hewlett-Packard Co.
4305 Cox Road
GLEN ALLEN, VA 23060
Tel: (804) 747-7750
A,C,E,M,P*

Hewlett-Packard Co.
Tanglewood West Bldg.
Suite 240
3959 Electric Road
ROANOKE, VA 24018
Tel: (703) 774-3444
C,E,P

Washington
Hewlett-Packard Co.
15815 S.E. 37th Street
BELLEVUE, WA 98006
Tel: (206) 643-4000
A,C,CM,E,M

Hewlett-Packard Co.
708 North Argonne Road
SPOKANE, WA 99212-2793
Tel: (509) 922-7000
C

West Virginia
Hewlett-Packard Co.
501 56th
CHARLESTON, WV 25304
Tel: (304) 925-0492
A,C,M

Wisconsin
Hewlett-Packard Co.
275 N. Corporate Dr.
BROOKFIELD, WI 53005
Tel: (414) 784-8800
A,C,E*,M

URUGUAY
Pablo Ferrando S.A.C. e I.
Avenida Italia 2877
Casilla de Correo 370
MONTEVIDEO
Tel: 80-2586
Telex: 802586
A,CM,E,M

Olympia de Uruguay S.A.
Maquines de Oficina
Avda. del Libertador 1997
Casilla de Correos 6644
MONTEVIDEO
Tel: 91-1809, 98-3807
Telex: 6342 OROU UY
P

VENEZUELA
Hewlett-Packard de Venezuela C.A.
3A Transversal Los Ruices Norte
Edificio Segre 2 & 3
Apartado 50933
CARACAS 1071
Tel: 239-4133
Telex: 251046 HEWPACK
A,C,CM,E,M,P

Hewlett-Packard de Venezuela, C.A.
Centro Civdad Comercial Tamanaco
Nivel C-2 (Nueva Etapa)
Local 53H05
Chuao, **CARACAS**
Tel: 928291
P

Albis Venezolana S.R.L.
Av. Las Marias, Ota. Alix,
El Pedregal
Apartado 81025
CARACAS 1080A
Tel: 747984, 742146
Telex: 24009 ALBIS VC
A

Tecnologica Medica del Caribe, C.A.
Multicentro Empresarial del Este
Ave. Libertador
Edif. Libertador
Nucleo "C" - Oficina 51-52
CARACAS
Tel: 339867/333780
M

Hewlett-Packard de Venezuela C.A.
Residencias Tia Betty Local 1
Avenida 3 y con calfe 75
MARACAIBO, Estado Zulia
Apartado 2646
Tel: (061) 75801-75805-75806-80304
Telex: 62464 HPMAR
C,E*

Hewlett-Packard de Venezuela C.A.
Urb. Lomas de Este
Torre Trebol — Piso 11
VALENCIA, Estado Carabobo
Apartado 3347
Tel: (041) 222992/223024
C,P

YUGOSLAVIA
Do Hermes
General Zdanova 4
YU-11000 **BEOGRAD**
Tel: 340 327, 342 641
Telex: 11433
A,C,E,P

Hermes
Titova 50
YU-61000 **LJUBLJANA**
Tel: 324 856, 324 858
Telex: 31583
C,E,M,P

Elektrotehna
Titova 51
YU-61000 **LJUBLJANA**
CM

ZAIRE
Computer & Industrial Engineering
25, Avenue de la Justice
B.P. 12797
KINSHASA, Gombe
Tel: 32063
Telex: 21552
C,P

ZAMBIA
R.J. Tilbury (Zambia) Ltd.
P.O. Box 32792
LUSAKA
Tel: 215590
Telex: 40128
E

ZIMBABWE
Field Technical Sales (Private) Limited
45, Kelvin Road North
P.O. Box 3458
HARARE
Tel: 705 231
Telex: 4-122 RH
E,P

64331-90903, SEPTEMBER 1985
E0985
Replaces: 64331-90902, September 1984
64334-90901, October 1984

HEWLETT
PACKARD

PRINTED IN U.S.A.