DATA COVERAGE PERFORMANCE EVALUATION
FOR REAL-TIME SYSTEMS

by

Gail Dean Brooks

Project submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Systems Engineering

APPROVED:  

D. C.S. Allison, Chairman

A. Dean  M. Nadler

April 1990

Blacksburg, Virginia
DATA COVERAGE PERFORMANCE EVALUATION
FOR REAL-TIME COMPUTER SYSTEMS

by

Gail Dean Brooks
Committee Chairman: Dr. Allison
Dean, Computer Science

(ABSTRACT)

Throughout the life cycle of a real-time, distributed computer system, numerous situations arise which require access to detailed system performance information. One important type of performance information is known as data coverage. Data coverage is the process whereby all ranges of values for a particular variable in a piece of software have been exercised. Data regarding variable values can be very useful in determining the correctness of program execution.

This project investigated the design, cost and usefulness of adding a data coverage performance evaluation capability within an existing Navy weapon system. The target system used is a real-time, loosely coupled distributed computer system. Experiments were created by writing programs in a high level language and executing the programs on the target system. Several of the programs contained "seeded" errors. Experiments were monitored and data was collected by an existing
performance evaluation system. The data was collected in a real-time, non-interfering manner and evaluated off-line. The off-line evaluation was accomplished by modifying an existing software package. The package did not include any capabilities for producing a data coverage report. Therefore, it was necessary to enhance the software package such that a data coverage report was created.

A description of the investigation and the various factors used for the evaluation of a data coverage capability, the design proposal, and the cost analysis are included in this report.
Table of Contents

I. EXECUTIVE SUMMARY ................................................. 1

II. INTRODUCTION ....................................................... 3
    A. System Performance Data ..................................... 3
        1. Reserved Capacity ...................................... 4
        2. Operating System Overhead .......................... 4
        3. Memory Utilization ................................. 4
        4. Resource Utilization ............................. 5
        5. CPU Throughput And Response Time .............. 5
        6. Program Path Coverage .......................... 5
        7. Operand Capture .................................. 6
    B. Statement of Problem ...................................... 6
    C. SCOPE OF PROJECT ........................................ 8
    D. PROJECT SCHEDULE ...................................... 9

III. BACKGROUND ..................................................... 10
    A. VEST Components ...................................... 11
        1. Event Trace Unit (ETU) .......................... 11
           a. Trace Software Addresses ................. 16
           b. Software Category Changes .............. 16
           c. Program Trace .............................. 17
           d. Program Coverage ........................... 17
e. Operand Capture  ........ 18

2. VEST Experiment Setup ........ 18

3. Data Reduction Software (DRS) .... 18

IV. PROPOSED CAPABILITIES OF DATA COVERAGE ............... 21

V. REQUIREMENTS FOR DATA COVERAGE REPORT GENERATION ........ 23

VI. FUNCTIONAL ANALYSIS ............... 25

VII. EXPERIMENT APPROACH AND DESIGN ...................... 26

A. Test Cases ............... 26

B. Execution Of Test Cases ........ 27

C. Report Generation ........ 30

D. Experiment Output ........ 30

VIII. EXPERIMENT RESULTS ...................... 33

B. Automation ............... 34

IX. HARDWARE DESIGN ...................... 36

A. Micro Sequencer Functions For A Data Coverage System ........ 37

B. Hardware Functions For A Data Coverage System ........ 38

1. Associative Memory And Match Logic (AMML) ............... 38

2. New-data Register ........ 38
3. Arithmetic Magnitude Comparator ........................................... 39
4. Main RAM ................................................................. 39
5. Hardware Controller ....................................................... 41

C. Report Generation Functions For A Data Coverage System ................................. 43

X. LIFE CYCLE ANALYSIS .................................................... 43

XI. CONCLUSIONS ............................................................. 45

XII. APPENDICES

Appendix A References
Appendix B Functional Analysis
Appendix C Test Program Source Files
Appendix D Log and Setup Files for Experiments
Appendix E DRS Input Files
Appendix F DRS Reports
Appendix G PDL for Microsequencer
Appendix H PDL for Report Generator
Appendix I Life Cycle Cost Analysis
Appendix J Acronyms
I. EXECUTIVE SUMMARY

This report details the investigation of a capability that, based on research, enhances the ability to achieve a high level of confidence in products produced for the target weapon system. This capability provides detailed system performance data information which has proven useful in determining the correctness of program execution.

The types of performance data that have been considered throughout this project are the range of values that a set of variables takes on during the execution of a piece of software and the truth table combinations for any logical expression. This type of performance data collection is referred to as data coverage. This project was concerned with evaluating the usefulness of a data coverage capability for use in the Quality Assurance group.

The Quality Assurance group is responsible for validating the operational software that is produced by an independent software development group. Currently, the software validated by the Quality Assurance group executes within one specially designed processor.

The investigation of a data coverage capability was accomplished by writing several test programs targeted for the specially designed processor. Several of the programs contained "seeded" errors to help prove the usefulness of data coverage. During program execution, data coverage information was collected by an existing performance monitoring system which contains a limited data coverage capability. The information collected was reformatted into a data coverage report by modifying an existing software package.
Even with the limited capabilities provided for the evaluation of a data coverage capability, results confirmed that a data coverage capability is a useful tool in the Quality Assurance environment. When used in conjunction with other tools, data coverage can enhance current Quality Assurance procedures for ensuring the correctness of a software system by providing more insight into program execution sequences.

Since a data coverage capability was deemed useful, a hardware design was created and documented by this report. The hardware is designed such that many different types of processors may be monitored for data coverage performance data gathering with the use of a POD. A transportable hardware design was decided upon because of the trend moving away from specially designed processors and moving towards commercial products. Current direction from Washington is that the specially designed processor will remain in the system and commercial processors will be "added" to the current system to build a more distributed system. Therefore, a standard interface becomes a necessity.
II. INTRODUCTION

One of the major functions of the work force with which the Quality Assurance group is a part of, is to design, develop, produce, verify, and maintain the operational software that executes on a real-time, loosely coupled, weapon system. This weapon system controls the launching of missiles.

Since this is a weapon system, high reliability is a requirement. The consequences of failure can be catastrophic. Normally when a malfunction occurs, the entire weapon system is put in a "down" state because of the uncertainties of continuing. In addition, failures occurring in a weapon system are much more difficult to troubleshoot for several reasons. First, weapon systems usually operate in an environment (e.g. submarine) far removed from the development facility. Therefore, problem reporting means passing of information from the operational environment back to the support facility. There can be oversights and misinterpretations. Secondly, weapon systems have to interface with nonstandard peripheral equipment such as navigation, launchers, etc. Such equipment usually has its own unique timing requirements. Problems that deal with timing constraints in a real-time system can be almost impossible to repeat. In view of this discussion, the need for non-interfering system performance data monitoring is imperative.

A. System Performance Data

There is a need in every phase of the life cycle of a real-time system for gathering detailed system performance data. Several factors are used in assessing overall performance of a system. Some general
ones are:
reserved capacity (idle time),
operating system overhead,
memory utilization,
resource utilization
CPU throughput and response time
program path coverage and data coverage

1. Reserved Capacity

In most systems, reserved capacity and idle time are closely related. However, there is an important distinction to keep in mind. If the system is idling because either all jobs have completed, or are waiting for reasons other than resource contention, then idle time reflects reserve capacity. Idle time caused by resource contention does not necessarily indicate reserve capacity since this idle time may not be usable unless the contention is resolved. In this case, there are usually methods to eliminate such contention once the cause has been accurately pinpointed. In such instances, that idle time becomes either absorbed by the existing jobs or becomes truly reserve capacity.

2. Operating System Overhead

A very popular measure in system performance evaluation is the amount of CPU resource consumed by the operating system. If this becomes a considerable amount, fewer resources are available for application programs which carry out the operational functions.

3. Memory Utilization
The portions of memory that are being used can be ascertained by analyzing references to all memory locations. This information can lead to more effective allocation and utilization of memory. When the frequency of reference to all memory locations is examined, conclusions can be drawn about whether it is appropriate to relocate certain programs on the mass memory devices and load them "as needed". In addition, knowing the access time of the main memory, it may be found that incorporation of a faster memory technology can make performance improvements. Looking at the frequency of use of memory for instructions, the most utilized program areas can be determined. These, in turn, can be analyzed for greater efficiency.

4. Resource Utilization

All computer systems contain a set of resources which, when used properly, will maximize performance. Among these are: memory, the CPU, mass storage devices, specialized I/O devices, and software resources. Measurement of the use of these resources can provide keys to how efficiently the computer system is being used to perform its functions.

5. CPU Throughput And Response Time

The application package for an embedded computer system consists of a set of jobs which must meet system throughput and response time requirements. The throughput requirement implies that the set of jobs must be executed at a certain minimum rate or the system fails. The response requirement implies that within a certain predetermined time constraint, a job must produce the desired result. Failure to meet either requirement indicates that the overall system is not performing its mission objective.
6. Program Path Coverage

Program path coverage is the process in which all paths throughout a piece of code are executed. This is a very reliable way of evaluating the correctness of a system, especially when used in conjunction with a data coverage capability. Unfortunately, in most cases, this is not feasible because the number of paths is infinite.

7. Operand Capture

Operand Capture is the process in which the data associated with the execution of an instruction or the access of a memory location is recorded. It was decided through this project that the operand capture capability could be used in the investigation and evaluation of a data coverage performance monitoring tool.

B. Statement of Problem

This report focuses on expanding the understanding of tools and methodologies, through the study of a data coverage capability, to enhance the techniques used to verify operational software produced for weapon systems. The verification of software is a vital part of the life cycle, especially since it accounts for 30 to 50 percent of the total development effort [13]. One of the most extensively used techniques in the Quality Assurance area is the decision-to-decision path coverage report. This report documents the fact that a branch has occurred or has not occurred. There is neither ordering nor number of occurrences implied in the report.

Through recent research efforts, it has been found that decision-to-decision path coverage may not be the best choice for use in the verification of real time software. William E. Howden, in his IEEE
paper "Theoretical and Empirical Studies of Program Testing" [1],
concludes that total path coverage is one of the most reliable ways of
testing, but impossible to obtain in most circumstances; and branch,
or decision-to-decision path coverage testing is much less
efficient at detecting errors. He bases this on the fact that most
errors are a result of executing a particular combination of paths.
Therefore, it becomes a "hit and miss" situation in branch testing.

Howden applied several testing methods to a set of programs
containing known errors. The programs were written in several different
languages and were of various sizes. Howden indicates that value
tracing and special values testing (data coverage) was responsible for
finding the greatest number of the errors in the larger programs in
which only one type of testing technique was used. Based on his
results, data coverage is a promising tool in the evaluation of the
correctness of programs. This is not to say that data coverage is
the only method that should be used. In fact, Howden indicates that
combined testing techniques is the ultimate way of testing. Combined
testing revealed almost ninety percent of the errors in the programs.
Data coverage is one of the combined techniques that was used.

As a result of research and evaluation efforts by this project,
data coverage analysis has been identified as a possible tool that
could better ensure the quality of a piece of software. Currently,
the quality of a piece of a software is evaluated by creating and
executing tests on the software. The question that arises is how is
the quality of the tests evaluated? If a test case finds an error in
the software, it is considered a good test case. But, what about a test
case that does not find any errors? Certainly it is not considered a bad test case just on that fact alone. In fact, there are lots of good test cases that may never find an error. Data coverage could aid in the answer to this question. For example, the evaluation of the test can be enhanced by performing data coverage during the execution of test procedures. If a test procedure causes new values of a piece of data to be recorded, then the test procedure has provided new information for the Quality Assurance group to analyze.

C. SCOPE OF PROJECT

The scope of this project is to evaluate the utility of a data coverage capability within the Quality Assurance environment. Data coverage, in the context of this report, entails two types of performance monitoring. This first is the process whereby all ranges of values for a particular variable have been exercised. This is referred to as the data range capability. The variables of interest are ones which appear in any type of conditional statement. For the high level language that is used to develop software for the real-time distributed computer system, this would entail the statement types IF, CASE, and conditional loops. The second type of data coverage monitoring is set of truth table information for any type of logical expression. The area of interest here is reversed from the data range capability. The data range capability is concerned with the final value of an expression whereas the truth table information is only concerned with the values going into making up the final value.
If it is found that data coverage is a worthwhile capability, then a possible implementation will be developed to incorporate a data coverage capability into the Quality Assurance environment. These goals will be accomplished in the following ways:

- design test programs that will best evaluate a data coverage capability, which may involve "seeding" of errors,
- use existing Verification and Evaluation System (VEST) to collect data for experiments,
- run data coverage collection experiments on several test programs by executing the programs in the real-time distributed system,
- generate reports on the experiments by writing a report generation package for data coverage or trying to modify an existing report generation package that is not set up for data coverage reporting capabilities,
- evaluate and document the results,
- if evaluation is favorable, translate the data coverage process into a system design,
- report conclusions and recommendations.

D. PROJECT SCHEDULE

The following is the schedule for this project.

January 22, 1990 – February 2, 1990 Design
February 5, 1990 – March 16, 1990 Code and Test Case Generation
March 19, 1990 – April 20, 1990 Hardware evaluation, Final Report, and Presentation Preparation
To better understand the current environment for the evaluation of a data coverage capability, some background on existing capabilities is provided.

III. BACKGROUND

The Computer Verification and Evaluation System (VEST) [4],[5], is an on-line, real-time, non-interfering data collection system used extensively by program developers and the Quality Assurance group to debug, test, and verify the software which executes on a real-time, loosely coupled distributed computer system. The main computer in the system is a specially designed computer for use in a weapon system environment. It is a single address, two state, general register, micro-programmed virtual machine. It has a priority interrupt structure which may be changed only in privileged state. The basic word length is thirty-two bits.

The evaluation system for this weapon system, called VEST, consists of several hardware and software components which interact to produce statistical reports about the execution of the software within the real-time distributed computer system. The components of the VEST system are shown in Figure 1.

The reason the VEST system can perform non-interfering data collection is that the computer it is collecting information from has been designed to send data across an interface for every type of instruction that is executed (see Figure 2). This interface is referred to as the Performance Evaluation Interface (PEI). The information is collected in real-time by the VEST system. The type of information
that is sent across the interface is:
- Program Status Word
- Contents of Arithmetic Logic Unit
- Instruction Buffer Register
- The 32-bit single word data value associated with the current instruction.

These quantities are provided by the hardware. The data is gathered by the hardware during instruction executions, tagged for identification, and relative time purposes, and transmitted across the PEI interface. The PEI outputs are issued at the computer’s minor cycle rate (4 MHz). Control of the interface is accomplished by means of a 5-bit microinstruction word field, referred to as the PEMX field. Figure 3 shows the data flow involved in the operation of the interface.

A. VEST Components

1. Event Trace Unit (ETU)

The ETU is a specially designed piece of auxiliary equipment capable of recording user selected activity in the real-time computer system [11]. Users are able to supply initialization parameters which define the activity of interest. Based on the data being monitored by the ETU, it captures the requested events. Event tracing is accomplished in real time without affecting the operation of the real-time computer system in any way.

Each event recorded consists of an event code (five bits), the CPU state (one bit; privileged or non-
VEST II

TRIDENT II
FIRE CONTROL SYSTEM
MK 98/1

TDCC DATA
FCS EVENT DATA

VEST II
EVENT TRACE UNIT

CM

TS SYNCH

trace
activation control

C/D

S/S DATA
S/S DATA
S/S DATA
S/S DATA

SUBSYSTEM EVENT TRACE UNIT

CM

C/D

DATA COLLECTOR

PERFORMANCE DATA FILE

PRINTED REPORTS
PLOTTED REPORTS

EXPERIMENT INITIALIZATION & CONTROL

DATA REDUCTION SOFTWARE

Figure 1
## Table 1

**PEI EVENTS**

<table>
<thead>
<tr>
<th></th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>10</td>
<td>EOR (31-16)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EOR (31-16)</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>PEI (31-16)</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>PEI (31-16)</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>PEI (31-16)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>PEI (31-16)</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>PEI (31-16)</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>EE (15-0)</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>VA</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>VA</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>ADDR</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>VALUE (31-16)</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td>DATA</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td>DATA</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td>DATA</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td>DATA</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td>TID MID</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td>TID MID</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>CM DATA</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ST** = TIP STATE (privileged or nonprivileged)

**TS** = TIME STAMP VALUE
privileged), a time stamp (TS) value (10 bits), and when necessary, one or two additional 16 bit words defining the data associated with that event. Table 1 lists the events that can occur along with any associated data. The TS value is used to define the time at which the event occurred. This value is taken from a 10 bit TS counter which counts up at user selected nominal frequencies of 4MHz, 1MHz, 100 KHz, 10KHz, 1KHz, 1Hz, or 0.1 Hz.

a. Trace Software Addresses

A major feature of the ETU is the ability to trace addresses in the system and application software packages. Tracing software addresses means that the ETU can ascertain whether specific addresses are encountered in the execution path. Addresses may be virtual or physical, and represent the location of code(instructions) or data values. Virtual addresses are the addresses that the programmer sees when examining a program listing at the assembly language level. A physical address is any location within the main memory unit of the computer.

The ETU implements the software trace capability by use of an Associative Memory (AM) commonly referred to as a "Content Addressable" memory. As the name implies, the AM is addressable by contents. It can be preloaded by the user with a set of patterns. During execution of the software, addresses of code being executed and addresses of data values being manipulated are continuously sent to the AM. If the AM responds with a match, then the activity of interest has occurred. The location in the AM which yielded the match along with the value of the time stamp counter are recorded.
b. Software Category Changes

The real-time computer system has two CPU states, privileged and non-privileged. Each program within the system has a unique task identifier and each module within the program has a unique module identifier. These are called software category changes. All such software category changes are capable of being detected by the ETU. When such a change is detected and the user has requested to be informed of such, an event is generated and recorded.

c. Program Trace

Any program can be traced if the source and destination of each branch can be dynamically ascertained. This is the precise program trace feature designed into the ETU. It is capable of generating the source branch (the location branched from) and the destination branch (the location branched to) for each branch in the executing software. Admittedly, this can generate voluminous amounts of data. For this reason, qualifications are added to this feature which filter the data. These qualifications permit this feature to be activated when any combination (including none) of the following conditions hold:

- CPU state is either privileged or non-privileged,
- A specific task is executing, or
- A specific module is executing.

d. Program Coverage

To facilitate program coverage, the ETU is augmented by the Intrinsic Program Coverage Unit (IPCU). The IPCU is able to detect which instructions were executed and indicate path departure data.
(whether the instructions were branched from or sequenced from).

e. Operand Capture

The ETU permits capture of the results of execution of certain instructions and some operands accessed from memory. The PEI provides such quantities and the ETU is capable of dynamically capturing them. Capture of such quantities is initiated by the AM match. The ETU has the ability to capture the results produced, or values manipulated, by single word load and store instructions.

2. VEST Experiment Setup

The user pre-defines the information that will be collected via a software initialization program [10]. The setup program transforms the information into a given protocol and transfers the information to the VEST hardware. Data collection is then ready to begin.

Once an experiment is started via user directive, VEST begins using the setup information in the following way. As data comes across the PEI interface, the hardware compares it to the user initialization data to determine if it is to be saved or thrown away. At the completion of the data collection process, the saved data is stored in a file which can then be used by the off-line report generation component of the VEST system, DRS [2].

3. Data Reduction Software (DRS)

The Data Reduction Software produces reports which accurately reflect the performance attributes of the computer. The raw data collected, along with user defined directives, define the inputs to DRS. These inputs specify the type of report to generate and the data
to be included in the report. The DRS operating environment is depicted in Figure 4.

Several standard printed report formats are available. Some of these are:

- **Event Listing** - A printout of events of interest in the order of occurrence,

- **Summary** - Supplies appropriate statistics about the occurrence of events for the requested report type,
DRS OPERATING ENVIRONMENT

- USER DIRECTIVES
- PDV
- SYMBOLIC ADDRESS MAPPING (SCHEMA)
- DECISION TO DECISION PATH (DDP)

DRS
PERFORMANCE EVALUATION SOFTWARE

- PRINTED REPORTS
- PLOT FILE
- PERFORMANCE SUMMARY FILE

FIGURE 4
- Time lines - characterizing system activity showing system attributes,

- Procedure Coverage - provides an analysis of the procedures covered during the experiment and a list of the procedures not covered,

- Decision-to-Decision path coverage - one of the most extensively used reports which indicates which path is taken (true or false) for each of the decision points within a program.

There is a possibility that only a subset of the data collected will be used in the report generation process. One nice feature of DRS is that it does not corrupt the file where the "raw" data is stored, it only reads the "raw" data for processing to produce the results for the reports. The benefit is that all the collected "raw" data remains in the file for future use.

There are times when a user wants to see cumulative results for several sets of data. For example, in the decision-to-decision report, a user may want to know what areas of code have not been executed after several test sessions. Cumulative reports are a very simple way of providing this capability. Therefore, DRS provides the capability to insert new results into the existing results, thereby providing a cumulative type of capability.

IV. PROPOSED CAPABILITIES OF DATA COVERAGE

The goal is to be able to capture all operands: double and single word; memory and register.
The type of information requested for each variable being monitored is:
- the minimum value attained by the variable,
- the maximum value attained by the variable,
- the value closest to zero (both higher and lower),
- the value closest to a user-defined value (both higher and lower), and
- an indication of whether the variable attained the user-defined value,

- an indication of whether the variable attained the value zero.

In addition, a combinational data coverage capability is desired. Combinations of variables used in logical expressions are of interest to the Quality Assurance group because they are used extensively throughout the controlling portions of the real-time operational system. These logical expressions are referred to as boolean equations. For example, the expression "(A AND B OR C)" is a boolean equation with 3 variables; A, B and C. The possible number of combinations for the three variables is 2 to the 3rd power, or 8. The combinations are listed below.

**Sample Truth Table**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
0  1  0
0  1  1
1  0  0
1  0  1
1  1  0
1  1  1

Therefore, a truth table data coverage capability is desired. In the example above, the truth table report will indicate how many of the 8 combinations of the 3 variables have been hit.

The user is responsible for setting up the experiment and the variables to be captured. The setup procedure will, at first, be a manual process whereby the user will be required to input each variable to be monitored, along with each statement that references the variable. This, of course, would not be acceptable in the actual system. Setup automation will be a requirement.

V. REQUIREMENTS FOR DATA COVERAGE REPORT GENERATION

The following requirements are satisfied currently with the Data Reduction Software (DRS):

- DRS interprets an existing Performance Data Volume (PDV) file format shown currently collected and written by VEST.
- DRS has the capability of mapping addresses of variables to addresses of instructions that use these variables. This is how the report will determine the relationship for the value of a variable and the instruction that generated that particular value. This is needed for a minimum/maximum report and
the combinational report.

- DRS has the capability to save the results from run to run. Previous results may be merged with succeeding runs of the program in such a way as to distinguish new coverage from previous coverage and report this information along with a summary for the total coverage from all the data, old and new. This capability will provide a means to evaluate test cases with respect to new "hits" from experiment to experiment. This should reduce duplication of test cases.

The following requirements will have to be satisfied to support a data coverage capability:

- The report must contain the lowest value of the operand found during the experiment along with the program instruction address that caused the lowest value to occur.

- The report must contain the highest value of the operand found during the experiment along with the program instruction address that caused the highest value to occur.

- The report must contain the two values closest to zero (above and below).

- The report must contain the two values closest to a user defined value (above and below).

- The report must contain an indication of whether or not the value ever matched the user defined value, if one was defined.

- The report must contain an indication of whether or not the value zero has been obtained.

- The boolean report deals with reporting how many combinations of the variables at any given location have occurred. This is in
response to the fact that the main controlling procedure in the real time system being dealt with is entirely made up of boolean computations. This procedure executes at a rate of eight times a second (every 125 ms.). The result of these computations drive the rest of the system. An example would be:

\[
A = B \text{ OR } C \text{ AND } \neg D
\]

The on-line data capture capability will capture the values of B, C, and D each time the above line is executed. The data coverage report would indicate the combinations that had been covered during the experiment. Notice here that the emphasis is not on the value of A, but on the combination of values of B, C, and D. In fact, the value of A will not even be reported in the combinational coverage report, unless A is used on the right hand side of an equation somewhere else within the program.

- The report must be able to map a virtual or physical address to a symbolic variable name and a symbolic line number in the high level language source file.

VI. FUNCTIONAL ANALYSIS

Basically there are three modes of operation for functional analysis purposes. Mode 1 contains all the functions necessary to prepare the VEST system for data collection. Mode 2 contains all the functions necessary for the data collection process itself. Mode 3 contains all the functions necessary for the data reduction and report generation parts of VEST. Appendix B shows the functional analysis diagrams for a VEST system with the Data Coverage Capability.
VII. EXPERIMENT APPROACH AND DESIGN

The capability for the current system is restricted to single word, memory accessed operands. Several test cases are generated to test the data coverage system with this restriction in mind. After each test case has been executed and the data has been collected, a report is generated. The approach for the report generation will be to use the current reporting capabilities of DRS. DRS does not currently support data coverage reports, but, through the use of the programming capabilities for the input specification to DRS, a data coverage report will be generated.

A. Test Cases

Several of these test cases will be designed such that errors are "seeded" into the code in order to help evaluate how the data coverage system can detect certain types of errors. These "seeded" errors were decided upon by researching past experience on the types of errors that have been very difficult to locate due to nonexistent tools that would allow them to be detected. For example, an array is declared to have twenty-four elements and is indexed from 0 to 23. It is used in a loop that goes from 1 to 24 and the loop counter is used as an index into the array.

Normally, the array bounds checking feature of the compiler will catch this type of error by creating code that checks the array bounds each time an element in the array is accessed. However, this takes about four machine instructions for each access. The real-time system being used as the environment for this project has a physical
main memory limitation of 256K 32-bit words. Because of this limitation, programmers are allowed to turn the bounds checking feature off to make the programs fit. So, in the situation described above, the program would destroy the contents of the memory location immediately following the array when the loop counter reaches the value of 24. Can this error be readily detected with the aid of data coverage? This type of question will be answered through the study.

The test cases are written in a block oriented, ALGOL-like high level language called THLL [3]. It is the same language that is used in the development environment for the real-time computer system. A list of the test programs is given in Appendix C. The purpose of each test program is described below.

Program 1- Common programming error where the arithmetic "=" is used instead of the logical "EQL". The result of this is that the false path will never be taken.

Program 2- Illustrates the used of a loop counter and an array.

Program 3- Illustrates the normal use of logical expressions in an IF statement.

Program 4- Illustrates a normal type of boolean expression.

Program 5- Illustrates the programming error where an array subscript limit is less than the limit of the loop initializing the array and the loop counter is being used as the array subscript.

Program 6- Illustrates a rather lengthy set of boolean equations. It also calculates boolean expressions within a loop using an array as the boolean operands.

B. Execution Of Test Cases
The programs are run through the THLL compiler [6] and THLL loader [7] to produce a load file. They are then run through a program which rearranges and reformats the information to be loaded onto the mass memory of the real-time system.

The approach is to run the test cases in a "standalone" (no operating system) environment. This decision comes from the fact that loading time for a "full" system is about twenty minutes. This is a constraint for two reasons. There is only one weapon system for use by the entire development and Quality Assurance teams. Therefore, each person must request time slots each week in order to run on the system. Due to the accelerated and tight schedules at the moment, this project was given very low priority and very small time slots, so a decision had to be made as to the quickest way of getting the experiments executed.

The fastest way for a program to gain control is from the bootstrap logic. Since the program is gaining control from bootstrap, no other program loading is necessary. This cuts the load time from twenty minutes to about one minute. However, this does put a constraint on the types of tests that can be executed in this environment. For example, no operating system calls can be made. But, even with the constraints being considered, it was decided that the worth of a data coverage capability could be obtained in this environment. Therefore, the program that gains control on bootstrap in the real-time computer is replaced with the test program. This involves some coordination with header and size information since the bootstrap program was written for a very specific application.
After the program is loaded, the execution is very simple and instantaneous. Debug and checkout of programs is enhanced by the use of a VAX based Software Development System (SDS) which has a dedicated link to the real time computer system. The SDS can control the execution of programs in the real-time computer via interactive commands from the user [8]. This is a very efficient and convenient way to checkout programs which are not affected by off-line interfering behavior. Once the test program is on the mass memory system, a simple boot command from the user loads the program into main memory. A sample log file from one of the experiments is shown in Appendix D. Also, as part of Appendix D, a list of the setup files is shown for each of the test programs. These setup files can be created off-line using an editor and run automatically once inside the SDS program. This is a very convenient and efficient way to do experiment setup.

The first and last instruction of each test program is replaced by a halt instruction with a unique identifying code. In this way, the program will halt when it first gets control from bootstrap. The user simply gives the "go" command and the program will execute until the second halt instruction is reached. The second halt instruction replaces the return instruction. This is necessary since bootstrap never expects to regain control and, in fact, if it ever does, it causes the processor to go into a software fail mode. The program can be run over and over again just by resetting the PSW register. This allows data values to be changed from run to run, which facilitates the data coverage information being collected. A sample scenario can be seen in
the log file in Appendix D.

C. Report Generation

The data coverage information is collected using the current VEST system. The VEST system has a software interface for experiment setup. The user simply types in the information to be recorded and starts the experiment. Once the experiment is started, the program can execute any number of times and the data will be collected continuously. After the user is finished with the experiment, data collection is stopped and the information is saved in a file for report generation.

The current report generation software package does not support data coverage reports. It does, however, support a user interface where the user can define the type of report to print out. Two types of reports are required to analyze the data coverage capability. The first is a data range report which will determine the minimum, maximum type values a variable has obtained. The second type of report is the combination report which will print out a truth table for any set of variables in a program. Both report input files are shown in Appendix E.

D. Experiment Output

Several sets of report runs are shown in Appendix F. Each report shows the data associated with an Associative Memory (AM) location. It is the Associative Memory that is used to set up the addresses for which data collection is to occur. Each report is explained below.

Report 1 corresponds to Program 1 described above. This is a data range report which reports on the range of values found for the
variables X and Y.

"AM 0" corresponds to X.
"AM 1" corresponds to Y.

Report 2 corresponds to Program 2 described above.
"AM 0" corresponds to the loop counter "I".
"AM 1" corresponds to the array "NUM" as a single entity.

Unfortunately, at this time, there is no way to distinguish between different elements in the array when they are referenced with a variable subscript. The maximum value of the loop counter is naturally going to be one more than the maximum value of the array subscripts due to the nature of loop limit checking.

Report 3 corresponds to Program 3 described above.
"AM 0" in the output file corresponds to "A".
"AM 1" corresponds to "X".
"AM 2" corresponds to "C".
"AM 3" corresponds to "Y".

This report demonstrates the case where multiple runs are performed. The data was altered at the beginning of each run to show the cumulative output shown in the report.

Report 4 corresponds to Program 4 described above. This report is an example of the combination report form for boolean type expressions.

"AM 0" corresponds to "A".
"AM 1" corresponds to "X".
"AM 2" corresponds to "C".
"AM 3" corresponds to "Y".

The truth table is printed such that a non-zero value is represented by a "1" (true) and a zero value is represented by "0" (false). This was done for uniformity and readability purposes. It was felt that a table of the actual values for the variables would be more confusing than beneficial. Ultimately, a table with "T" and "F" is desired. DRS does not currently support a character output format.

Report 5 corresponds to Program 5 described above.

"AM 0" corresponds to "I".

"AM 1" corresponds to "NUM" (any value in the array).

"AM 2" corresponds to "NUM(23)".

"AM 3" corresponds to "NUM(0)".

This report is an attempt to show the programming error where the loop counter does not match the array subscript definition. It is very hard to see the error at this level. A symbolic capability would be a necessity in order to easily detect this type of error. If the user looks very closely at the output, it can be determined that the largest value reported to be in the array is 24, but when the last array element is referenced, it has a value of 23. This should indicate to the user that something is incorrect and requires further investigation.

Report 6 corresponds to Program 6 described above. This report shows several truth tables. The input file to the DRS program was made up very specifically for this program. That is why the variable names are printed with each of the truth tables. In actuality, the user would not have time to do this for every program. Therefore, there
is a need to be able to create this information automatically.

VIII. EXPERIMENT RESULTS

A. Limitations

First, only 32-bit single word operands can be collected. The limiting factor here is that all floating point numbers are 64-bits and names within the system are defined as eight characters which makes them 64-bits also. Secondly, operands are only sent across the interface for store and load type instructions. This is a very limiting capability, especially when one of the key areas of interest is compare instructions (where branching can occur). However, the enhancements necessary to provide complete data coverage would require some ROM changes within the real-time distributed computer system and are, therefore, beyond the scope of this project.

Another limitation on the report generation side of the experiments is that variables at the virtual level or physical level cannot be mapped to the symbolic level at this time. A symbolic mapping capability does exist in the current VEST system, but it is only implemented for program coverage reports. Because of this, "seeded" errors may not always be easily detected. For example, an array is declared to have twenty-four elements running from zero to twenty-three. A loop is used to initialize the array and the loop counter runs from one to twenty-four. The report will show the values of the array at a virtual level and it will show the loop counter having a value of twenty-five (one more than the limit). But it is not intuitively obvious at this point that the loop counter does not match
the indices of the array. If the array could be printed in a symbolic manner, such as the element name followed by its subscript, then it would be more easily detectable that the last subscript exceeds the size of the array.

B. Automation

To set up a full data coverage experiment (where all data variables within a code segment will be captured) would entail a tremendous amount of effort given the current capabilities. Eventually, this process will be automated as much as possible. For example, the user may specify a program and the setup process will be intelligent enough to pick out all variables and all references to the variables automatically. It will then convert that information into a format such that the data collection process can perform its function. This will be done before the data coverage collection process begins. Once the data is collected, the user may again select a subset of the variables captured to produce reports. A suggested overall control flow of the Data Coverage System is shown in Figure 5. Several parts of the data coverage system are marked as "new/modified". The picture is intended to show how a data coverage capability could fit into the current VEST system. The "new/modified" components reflect the efforts of this
project.

There is a need to be able to map the occurrence of AM's to their symbolic reference within the program. This information may have to come from a separate file, perhaps made up by the compiler. This file would be used by the report generation process to print a symbolic representation of any data coverage experiment.

Based on these results, it was decided to develop an initial high level hardware design that supports the data coverage capability. The following presents this design.

IX. HARDWARE DESIGN

In this section there are two types of hardware level designs considered. The first deals with using a microsequencer to implement a data coverage capability. This would basically consist of a "down loadable" program which would control the data coverage collection process. The second deals with using a hardware comparator. Both designs assume the input to the data coverage system will provide sufficient address and data information for it to do its job. These designs are performed to give an overall "high level" picture of the types of functions that are needed to provide a data coverage report for use by the Quality Assurance group.

During the analysis of the two types of hardware design, two main issues were considered: flexibility and speed. The microsequencer controller gives a lot of flexibility whereas the hardware compare would be very inflexible. But, when dealing with speed, the hardware compare is much faster than the microsequencer
controller.

In view of the above considerations, the design will consist of a combination of a microsequencer and hardware comparator design. The microsequencer will provide the flexibility on the front and back ends of the data collection and the hardware comparator will provide the speed needed to keep up with the data coming from the target processor.

It is highly desirable that this design be transportable to any system that the Quality Assurance group is responsible for. The Quality Assurance group will soon be dealing with several types of processors. This has already been decided by the weapon system community. The basic trend is moving toward asynchronous distributed systems. In order to be prepared for this, it was decided that an in-circuit POD be added as a front end to the data coverage system.

Lastly, a decision had to be made about whether to add on to the current VEST system or build an entirely new "generic" Data Coverage System. Even though a lot of time and effort has been spent on the current VEST, the trend is moving away from only supporting very specific processors. Therefore, it was felt that a generic Data Coverage Unit was more desirable in hopes that it will be more compatible with future systems.

A. Micro Sequencer Functions For A Data Coverage System

The high level design for a microsequencer controller is shown in a Program Design Language (PDL) format in Appendix G. The basic functions to consider are the collection of data requested by the user in a real-time fashion, setting several flags for the hardware compare logic, and saving the data for report generation at the end of
each experiment.

B. Hardware Functions For A Data Coverage System

The design of a Data Coverage Unit (DCU) contains the logic necessary to perform the majority of the data coverage functions such as the bookkeeping function tacked onto the point where the match signal and the Associative Memory (AM) address are generated.

The design of the DCU is composed of five main parts:

- Associative memory and match logic (AMML)
- New-data register (NDR)
- Arithmetic magnitude comparator (AMC)
- Main RAM (MR)
- Hardware controller (HC)

1. Associative Memory And Match Logic (AMML)

The AMML will contain \(2^{*n}\) words (2 to the nth power). Each word must be wide enough to express the computer’s address of the variable of interest. It will produce a match signal and an address. Input to the AMML will consist of computer addresses which will be sent by the microsequencer front end.

2. New-data Register

The new-data register is a 64-bit register. It is a bank of D-type edge-triggered flipflops with tri-state outputs. The path to the new-data register will be masked to allow only certain types of information to enter, such as ALU data. This masking will be done by the microsequencer front end. The input data path is a 32-bit wide bus which is multiplexed into two sets of 32-bit wide paths. This configuration is necessary as input data will occur as one
or two 32-bit words (integers and floating point numbers). Thus, the least significant 32-bits will be clocked into the register, the paths multiplexed, and the most significant 32-bits clocked in. The multiplexer select, new data register load, and a data-loaded signal will be generated by the Microsequencer controller.

3. Arithmetic Magnitude Comparator

The comparator consists of nine 8-bit magnitude comparator chips arranged in such a fashion as to provide a full 64-bit magnitude compare. The inputs to the comparator come from the new-data register and the Main RAM. The suggested parts provide the three decisions (greater than, less than, equal to) in one chip.

4. Main RAM

The DCU must have its own R/W memory in order to store the preset variable values and each of the five desired results. This memory must have as many words as does the AM (2**n) with a one-to-one mapping between the addresses of values in both. For example, the variable to monitor is found at a unique address X. Put the address X in the AM at AM address 256. The contents of main RAM address 256 contains the preset value to compare to the variable and the five data coverage output values corresponding to the variable. The main RAM is divided vertically into six pages, five of which are 64-bits and one of which is 1-bit. Four of the 64-bit pages store the four 64-bit wide coverage outputs while the 1-bit page stores the "equal to" data coverage output. The other 64-bit page stores the preset constant. The pages may be symbolized as shown in Table 2.

The addresses to the Main RAM come from the DCU. The read/write signal and RAM Page select field both come from
### TABLE 2

<table>
<thead>
<tr>
<th>Hex Am Address</th>
<th>MAIN RAM WIDTH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;1 bit&gt;</td>
</tr>
<tr>
<td>00</td>
<td>Page</td>
</tr>
<tr>
<td>01</td>
<td>HIT</td>
</tr>
<tr>
<td>02</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 3

**PIPELINE BIT DEFINITION**

<table>
<thead>
<tr>
<th>AM MATCH</th>
<th>DATA LOADED IN NEW-DATA</th>
<th>P &lt; Q</th>
<th>P = Q</th>
<th>P &gt; Q</th>
<th>MAIN RAM R/W</th>
<th>RAM SELECT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
the DCU microsequencer. Also, a 64-bit path must exist from the new-data register to the data-in lines of pages MAX, MIN, HI, and LO. This path is used to update the RAM values as new extremes are detected. VAL does not need a path from the new-data register. Since HIT is a one-bit field, it must be initialized to zero at the start of a test session and set only if the new-data is equal to the VAL value. This effect is most easily accomplished when the Write signal is defined. If Write is active high, then the Write line is fed into the data-in path of the HIT page. If Write is active low, then the logical NOT (i.e. inverse) of Write signal with the HIT page selected sets the addressed location.

5. Hardware Controller

The hardware controller controls the sequence of signals necessary to perform the compares and "bookkeeping" actions. It has five different input signals and four output signals to accomplish its tasks. These signals comprise the DCU controller pipeline as shown in Table 3.

The following is a list of the signal names as presented in the pipeline definition and their meanings:

- **AM Match**: An input signal from the AM indicating that a match has been made in the AM. Pipeline bit 0.

- **Data Loaded in new-data**: An input signal from the microsequencer front end which tells the microprogram that a new variable value is available in the new-data register. The microprogram waits for this signal to follow an AM Match and then it initiates the comparisons of the new-data register value with values addressed in the
Main RAM. Pipeline bit 1.
- $P > Q$: A decision signal from the comparator. Pipeline bit 2.
- $P = Q$: A decision signal from the comparator. Pipeline bit 3.
- $P < Q$: A decision signal from the comparator. Pipeline bit 4.
- Read/Write: An output signal from the hardware controller which allows a Read/Write to be performed on the main RAM. Pipeline bit 5.
- RAM Select: An output signal from the microsequencer which allows the six separate pages of main RAM to be detected individually. Since there are six pages of RAM, this signal is a field composed of an encoded group of three bits. Pipeline bits 6, 7, and 8.

The RAM Select signal is an encoded 3-bit field with the following definitions:

- $\text{VAL} = 000$
- $\text{MAX} = 001$
- $\text{MIN} = 010$
- $\text{HI} = 011$
- $\text{LO} = 100$
- $\text{HIT} = 101$
- not used = 110
- not used = 111
An overall control flow for a Data Coverage Unit is shown in Figure 6.

C. Report Generation Functions For A Data Coverage System

Appendix H shows a high level design in a Program Design Language (PDL) for the report generation functions. These are the perceived functions for the data range capability and the combination capability.

X. LIFE CYCLE ANALYSIS

Appendix I shows an entire life cycle analysis for a Data Coverage System. The life cycle is broken down into three categories: Research and Development, Production and Construction, and Maintenance and Support.

The total life cycle cost is calculated at a little over $900,000.00. This cost is determined using a Present Cost Evaluation at an interest rate of 10%.
XI. CONCLUSIONS

The monitoring of real-time distributed computing systems involves the collection and interpretation of information collected in a real-time, non-interfering manner. This information can be used for evaluating the performance of distributed computing systems. Emphasis during this project was concentrated on data coverage. Two types of data coverage capabilities were evaluated. The first is called a data range capability. Reports were produced on several different ranges of a variable such as the minimum value and maximum value a variable has obtained. The second type of capability deals with a truth table report. Emphasis here is on logical expressions. The user wants to know how many of $2^n$ possible truth table combinations have been obtained for an "$n" variable logical expression. This is important in the Quality Assurance environment because the entire real-time system is controlled by logical expression evaluations.

Six separate test programs were created to run as experiments. Each one demonstrated something different dealing with data coverage. Two test cases were created for truth table combination reports and the four others were created to test the data range capability. Two of the data range test cases contained "seeded" errors. These "seeded" errors were picked because this project discovered that these particular errors would not be found using existing Quality Assurance capabilities.

The first "seeded" error could be found using path coverage or data coverage. It is an example of "unreachable" code. Data
coverage detected the fact that the variable always had a non-zero value which means the logical expression always evaluates the "true". Path coverage would determine that the "false" path never gets executed.

The second "seeded" error would not have been found with path coverage. This is the case when an array subscript runs from 0 to 23 and the loop counter used as the subscript for initializing the array runs from 1 to 24. This error was not easily detected in the current data coverage capability because of the lack of symbolic level reporting. The user had to look closely at the output and correlate it back to the symbolic level to detect the error.

Howden [1] showed that a data coverage capability can find sixty-one percent of the errors. Path testing can normally find sixty-four percent of the errors. A combination of path and data coverage testing can find as many as eighty-nine percent of the errors. These statistics have been calculated using programs with "seeded" errors. Therefore, a combination of path coverage testing and data coverage testing is highly desirable.

This project discovered that the current capabilities of the VEST system do not meet the requirements for a data coverage system due to limitations both in the VEST system itself and in the processor with which the VEST system is communicating. An explanation of these limitations is given for clarity. The Performance Evaluation Interface (PEI) only sends certain information across the interface to VEST. Two important pieces of information are not sent. These are floating point operands and RAM register operands. Therefore, the only operands that
can be collected are integer values coming from a main memory location accessed with a virtual address (RAM registers are accessed by register number). Also, these integer operands are only sent across the interface by the PEI during "load" and "store" instructions. Such instructions as "compare" therefore, do not send any data across. Unfortunately, "compare" instructions are of great interest to the Quality Assurance group when using a data coverage capability.

This project has shown that there is potential for the use of data coverage in the Quality Assurance world. It could be very beneficial when all the capabilities are provided. This means all types of operands (e.g. floating point, double precision integers, etc.) in conjunction with any type of instruction (current capability only sends data across for load and store instructions). Now that some of the limitations and enhancements have been defined, a greater emphasis can be made towards creating a data coverage system eliminating the limitations and incorporating the enhancements. Also, there must be a move to support several types of processors in view of the fact that the current system is not being replaced, but being added onto using more current technologies through the purchase of commercial products.

It was discovered, through a literature research that was done as part of this project, that there is a small amount of interest and research in the area of "real-time" data coverage capabilities [9]. Lots of systems exist that will do data coverage, but they all have one of two unacceptable constraints. They either collect data by controlling the execution of the software (stop and go) or they collect data "real-time" for a fixed amount of data collection memory (which has been found
to hold data for about 500 assembly language instructions, using a normal mix). Since the Quality Assurance group deals with real-time constraints and with systems much, much larger than 500 assembly language instructions, these constraints are unacceptable.

Research is ongoing in the Quality Assurance group to provide insight about performance monitoring capabilities in real-time systems in general. Data coverage is included as part of this research. It has been found that the faster technology gets, more and more information hiding occurs. This is making it very difficult to provide off-chip capabilities for performance monitoring [12]. Also, as a result of market surveys, commercial vendors seem to be very slow in showing interest in providing non-interfering performance capabilities. Also, vendors presently do not want to provide algorithms (such as pre-fetch queues) which would enable emulation systems to be developed.

The project has been very beneficial in that a useful capability has been investigated and proven useful. Several limitations were discovered (such as single word operands, limited instructions set for data collection, and no data coverage report software), but ways around them were found so that the project could continue. Results of the project will enable the Quality Assurance group to start using the data coverage capability to some extent. Hopefully, this will enable the group to start looking at new and better ways of doing business so that when the new processors are brought on board, the Quality Assurance group will be prepared to deal with them.
APPENDIX A

REFERENCES

July 1978.


APPENDIX B

FUNCTIONAL ANALYSIS
SUBFUNCTIONS
Ref. 3.1

Get user's inputs for data collection

3.1.1
Set up hardware signals to monitor

3.1.2
Set up memory addresses to monitor

3.1.3
Set up which programs to monitor execution

3.1.4
Set up memory addresses to perform operand capture

3.1.5
Set up which devices to monitor

Ref. 7.2

Compare data with user experiment setup information

7.2.1
Save data in file if it compares to user input

7.2.2
Throw data away if it doesn't compare
OPERATIONAL ACTIVITY
Ref. 6.0
Perform data collection

SUBFUNCTIONS
6.1
Accept data from PEI interface
6.2
Time stamp data
6.3
Send data to data coverage logic
6.4
Continue data collection until end of experiment
6.5
Maintain communication with user at software development system

or

End of experiment occurred
End of experiment occurred by user interactively
End of experiment occurred by stopping it
OPERATIONAL FUNCTION
Ref. 10.0
Generate information needed for reports

10.1 SUBFUNCTIONS

Program coverage data

10.2
Hardware events data

10.3
Device utilization data

10.4
Data coverage information

10.5
Send data to appropriate report generation section

10.1.1
Record beginning and ending addresses with associated times

10.1.2
Count the number of times an event occurred

10.2.1
Count the number of times an event occurred

10.3.1
Record percentages for each device

10.4.1
minimum maximum report info

10.4.2
value hit report info

10.4.3
combinational report info
OPERATIONAL FUNCTION
Ref. 11.0

Generate Reports

and

SUBFUNCTIONS

11.1
Format reports according to user specification

11.2
Perform statistical analysis on results

11.3
Send reports and analysis to user specified output device
Ref Fig 3,6.1
With system operating in mode 2, check REI interface function

Ref Fig 3,6.2
(mode 2) Check time stamping function

Ref Fig 3,6.4
Check data to storage medium function (mode 2)

Ref Fig 3,6.5
(mode 2) Check real time communication to user function

Reference function 13.0

(Page 1 of Figure 4)
14.0 Localize fault to software module

14.1 Make correction to module

14.2 Take faulty module out and insert new module in

14.3 Rebuild and deliver new system
APPENDIX C

TEST PROGRAM SOURCE FILES

tst1nopt
BEGIN

//priv
//SCHEMA
//OPT 0
GLOBAL tst1nopt;
/******************************************************************************/
/* */
/* The purpose of this test program is to show */
/* a common programming error. This program will*/
/* never execute the false path of the IF. */
/* */
/* The programmer has incorrectly used the */
/* arithmetic "=" when the logical "EQL" is */
/* needed. */
/* */
/******************************************************************************/
DEFINE LINK PROCEDURE tst1nopt;

BEGIN
/* LOCAL DATA */
OWN INTEGER A, B, C,
    I, /* LOOP COUNTER */
    SUM; /* CHECK SUM TO BE RETURNED */

INTEGER X,Y,Z;

INTEGER COMPONENT HALF1 (ARITHMETIC FIELD(0,16), OFFSET 0);
INTEGER COMPONENT HALF2 (ARITHMETIC FIELD(16,16), OFFSET 0);

/* PROCEDURE */
Y = 1;
IF (X = Y) THEN
  I = I+1
ELSE
  I = 0
ENDIF ;
END;
END FINIS
tst2nopt
BEGIN
\\priv
\\SCHEMA
\\OPT 0
\\bounds 0
GLOBAL tst2nopt ;
/*********************/
/_* /
/_*/
/_*/  The purpose of this test program is test data */
/_*/  coverage with an array and a loop counter. */
/_*/
/*********************/
DEFINE LINK PROCEDURE tst2nopt ;

BEGIN
/* LOCAL DATA */
OWN INTEGER A, B, C,
   I,           /* LOOP COUNTER */
   SUM;         /* CHECK SUM TO BE RETURNED */

OWN INTEGER ARRAY NUM(23) ;
INTEGER   X, Y, Z;

INTEGER COMPONENT HALF1 (ARITHMETIC FIELD(0,16), OFFSET 0);
INTEGER COMPONENT HALF2 (ARITHMETIC FIELD(16,16), OFFSET 0);

/* PROCEDURE */
FOR I = 0 STEP 1 UNTIL 23 DO
   NUM(I) = I
ENDDO ;
END;
END FINIS
tst3nopt
BEGIN
\priv
\SCHEMA
\OPT 0
\bounds 0
GLOBAL tst3nopt ;
/*
 */
/*
 This test program is designed to show the use of
 */
/*
 logical comparisons in an IF statement. The purpose
 */
/*
 of data coverage here is to provide information on
 */
/*
 how to manipulate the values in order to reach both
 */
/*
 paths.
 */
/*
*/
/*
*******************************************************************************/
DEFINE LINK PROCEDURE tst3nopt ;

BEGIN
/* LOCAL DATA */
OWN INTEGER A, B, C,
    I,       /* LOOP COUNTER */
    SUM;     /* CHECK SUM TO BE RETURNED */

OWN INTEGER ARRAY NUM(23) ;
INTEGER   X,Y,Z;

INTEGER COMPONENT HALF1 (ARITHMETIC FIELD(0,16), OFFSET 0);
INTEGER COMPONENT HALF2 (ARITHMETIC FIELD(16,16), OFFSET
/* PROCEDURE */

A = 1.5;
X = 1;
C = 2.5;
Y = 2;
IF (A GRT X) AND (C LES Y) THEN
    SUM = A + B
ELSE
    SUM = X + Y
ENDIF;

END;
END FINIS
tst4nopt
BEGIN
\priv
\SCHEMA
\OPT 0
\bounds 0
GLOBAL tst4nopt ;
/*******************************/
/*/                     */
/*/ The purpose of this test program is to show a */
/*/ boolean expression which will be used in the */
/*/ combination report generation part of data */
/*/ coverage. */
/*/ */
/*******************************/
DEFINE LINK PROCEDURE tst4nopt ;

BEGIN
/* LOCAL DATA */
OWN INTEGER A, B, C,
I,     /* LOOP COUNTER */
SUM;    /* CHECK SUM TO BE RETURNED */

OWN INTEGER ARRAY NUM(23) ;
INTEGER   X,Y,Z;

INTEGER COMPONENT HALF1 (ARITHMETIC FIELD(0,16), OFFSET 0);
INTEGER COMPONENT HALF2 (ARITHMETIC FIELD(16,16), OFFSET 0);

/*/ PROCEDURE */
A = 0 ;
X = 1 ;
C = -1 ;
Y = -3 ;
C = (A OR C) AND (X AND Y) ;
END;
END FINIS
tst5nopt
BEGIN
   \priv
   \SCHEMA
   \OPT 0
   \bounds 0
GLOBAL tst5nopt ;
******************************************************************************
/*
/* This program illustrates the programming error */
/* where an array subscript limit is less than the */
/* limit of the loop initializing the array and the */
/* loop counter is being used as the array subscript. */
/*
******************************************************************************
DEFINE LINK PROCEDURE tst5nopt ;

BEGIN
   /* LOCAL DATA */
   OWN INTEGER A, B, C,
      I,            /* LOOP COUNTER */
      SUM;          /* CHECK SUM TO BE RETURNED */

   OWN INTEGER ARRAY NUM(23) ;
   INTEGER      X,Y,Z;

   INTEGER COMPONENT HALF1 (ARITHMETIC FIELD(0,16), OFFSET 0);
      INTEGER COMPONENT HALF2 (ARITHMETIC FIELD(16,16), OFFSET 0);

   /* PROCEDURE */
   FOR I = 1 STEP 1 UNTIL 24 DO
      NUM(I) = I
   ENDDO ;

   IF NUM(23) EQ 0 THEN
      SUM = 0
   ENDIF ;
IF NUM(0) EQL 0 THEN
    SUM = NUM(23)
ENDIF;

I = SUM;
SUM = -1;
END;
END FINIS
test6nopt
BEGIN
\private
\schema
\opt 0
\bounds 0
GLOBAL test6nopt;
/**************************************************************************/
/* */
/* This program is designed to show a rather */
/* lengthy set of boolean equations, including */
/* a boolean equation within a loop using */
/* an array as the boolean operands. */
/* */
/**************************************************************************/
DEFINE LINK PROCEDURE test6nopt;
BEGIN
/* LOCAL DATA */
OWN INTEGER A, B, C, D, E, F, G, H, I, J, K, L, M, N,
     O, P, Q, R, S, T, U, V, W;

OWN INTEGER ARRAY NUM(23);
INTEGER X, Y, Z;

INTEGER COMPONENT HALF1 (ARITHMETIC FIELD(0,16), OFFSET 0);
INTEGER COMPONENT HALF2 (ARITHMETIC FIELD(16,16), OFFSET 0);

/* PROCEDURE */
PRESET BEGIN
A = 0;
B = 1;
C = 0;
D = 1;
E = 0;
F = 1;
G = 0;
H = 1;
I = 0;
J = 1;
K = 0;
L = 1;
M = 0;
N = 1;
O = 0;
P = 1;
Q = 0;
R = 1;
S = 0;
T = 1;
END; /* PRESET */

P R E S E T  N U M ( 0 )  T O  N U M ( 2 3 )  =
1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,
$X = 1;$
$Y = -3;$

$B = (A \text{ OR } C) \text{ AND } (X \text{ AND } Y);$  

$D = E \text{ AND } F \text{ OR } G;$

$H = I \text{ OR } K \text{ AND } L \text{ OR } M;$

$N = (O \text{ OR } P) \text{ AND } Q;$

$R = B \text{ AND } D \text{ AND } H \text{ AND } N;$

FOR $I = 1$ STEP 1 UNTIL 23 DO
    NUM($I+3$) = NUM($I$) ANDB NUM($I+1$) ORB NUM($I+2$)
ENDDO;

END;
END FINIS
APPENDIX D

LOG AND SETUP FILES FOR EXPERIMENTS

LOG FILE FROM PROGRAM 3 EXPERIMENT

GBROOKS 19-MAR-1990 14:40:33.73 ETS TADPOLE REV 2.004
******************************************************************************
This is the version dependent data file for the TADPOLE ETS system.
It contains information about the TRIDENT II Operating System.
*/
INIT VEST/TAD
LPR_PSDN  = 33
OS_PCB_ABS = 100
@
****************************************************************************** END OF TADPOLE INITIALIZATION DATA
*******************************************************************************/
* exec test3_setup.jou
/* BEGINNING OF SDSU1:[GBROOKS.DATA_COV]TEST3_SETUP.JOU; */
* conn bp
* conn dcp
* sess
SESSION NAME: test3
SESSION SEQUENCE NUMBER:
TFCS MODE:
TFCS SOFTWARE VERSION:
METU1 DEFINITION:
METU2 DEFINITION:
METU3 DEFINITION:
METU4 DEFINITION:
ADDITIONAL COMMENTS:
* am
AM EVENT IS DISABLED.
OC EVENT IS DISABLED.

QUALIFY OC ON:

AM NUMBER [/OC | /RESET]: 0/oc
AM CONTENTS: e'c011
SYMBOLIC NAME: a
ASSOCIATED COMMENT: /* this is 'a' */
ASSOCIATED COMMENT:

AM NUMBER [/OC][/RESET]: 1/oc
AM CONTENTS: e'c013
SYMBOLIC NAME: c
ASSOCIATED COMMENT: /* this is 'c' */
ASSOCIATED COMMENT:

AM NUMBER [/OC][/RESET]: 2/oc
AM CONTENTS: e'c017
SYMBOLIC NAME: x
ASSOCIATED COMMENT: /* this is 'x' */
ASSOCIATED COMMENT:

AM NUMBER [/OC][/RESET]: 3/oc
AM CONTENTS: e'c019
SYMBOLIC NAME: y
ASSOCIATED COMMENT: /* this is 'y' */
ASSOCIATED COMMENT:

AM NUMBER [/OC][/RESET]: @
* boot
TADP-E, Unexpected Stop on Bootstrap
* ins psw
  P'16' X'C000C001': @
* EXP
PDV FILE NAME:
TADP-I, Data Collection Started
/* END OF SDSU1:[GBROOKS.DATA_COV]TEST3_SETUP.JOU; */
* bre 0 e'c011
TYPE OF FETCH (INSTRUCTION or OPERAND): i
USE STATE (YES or NO):
TASK ID (0-255 or IGNORE):
EXECUTE FILE NAME:
BK NUMBER (number[/DEL] or */DEL): @
* go
* *** BP BREAKPOINT 0 OCCURRED AT E'C011' AT 14:41:33.15 ***
* ins e'e000
  E'E000'  X'00000000':
  E'E001'  X'00000000':
  E'E002'  X'FFFFFFF': 1
  E'E003'  X'00000000':
ADDRESS[/format]: e'e066
  E'E066'  X'00000001': 2
  E'E067'  X'FFFFFFFD': 3
  E'E068'  X'00000000': @
* go
*** BP HALT OCCURRED AT 14:43:30.74 ***
* PSW: C200C027  IBR: 52000DEF
* ins 16
  P'16'  X'C200C027': x'c200c000
  P'17'  X'00000000': @
* go
*** BP HALT OCCURRED AT 14:43:39.39 ***
*      PSW: C200C001   IBR:  52000ABC
*      go
*      *** BP BREAKPOINT 0 OCCURRED AT E'C011' AT 14:43:41.12 ***
*      stop
TADP-I, Data Collection Terminated
TADP-I, NUMBER OF 16-BIT REAL-TIME WORDS WRITTEN = 0000000F (hex)
SESSION TEST3 SEQUENCE NUMBER 1 HAS COMPLETED.
Reading Block starting at Word (16-bit) 00000000 (hex)
    PDV FILE   SDSU1:[GBROOKS.DATA_COV]TEST3.PDV;9 HAS BEEN CLOSED.
*      exit
PROGRAM 1 SETUP

sess
testl

am

0/oc
e0'c00e,0,1
operandy
/* this is the loading of y */

1/oc
e0'c00f,0,1
operandx
/* this is the storing of x */

@
LOG AND SETUP FILES FOR EXPERIMENTS

PROGRAM 2 SETUP

conn bp, dcp
boot
ins psw
@
expe

conn bp, dcp
sess
test2

am

@/oc
e`c014
loadi
/*! this is the loading of I, the loop counter */
@
boot
ex 16
ins psw
@
ins ibr
@
expe

go
stop
LOG AND SETUP FILES FOR EXPERIMENTS

PROGRAM 3 SETUP

conn bp
conn dcp
sess
test3

exec vam.jou
am

0/oc
e'c011
a
/* this is 'a' */

1/oc
e'c013
b
/* this is 'c' */

2/oc
e'c017
x
/* this is 'x' */

3/oc
e'c019
y
/* this is `y` */
@
PROGRAM 4 SETUP

exec vam.jou
conn bp
conn dcp
sess
test4

am

0/oc
e'c011
a
/* this is 'a' */

1/oc
e'c013
x
/* this is 'c' */

2/oc
e'c017
c
/* this is 'x' */

3/oc
e'c019
y
/* this is 'y' */

@
RESET
boot
ins psw
@
EXP

exec vam.jou
am

0/oc
e'c014'
i
/* loop counter I */

1/oc
e'c012'
num
/* the array num gets stored */

2/oc
e’c017’
um23
/* line 28 of tst5nopt */

3/oc
e’c01a’
um0
/* line 32 tst5nopt */

4/oc
e’e004’
sum
/* data address of sum */

@
conn bp, dcp
boot
ins e’c000
@
PROGRAM 5 SETUP

sess
test5

exp
PROGRAM 6 SETUP

conn bp,dcp
exec vam.jou
sess
test6

am

0/oc
e'c00f
a
/* line 53 tst6nopt */
/* A */

1/oc
e'c011
c
/* line 53 tst6nopt */
/* C */

2/oc
e'c015
x
/* line 53 tst6nopt */
/* X */

3/oc
e'c017
y
/* line 53 tst6nopt */
/* Y */

4/oc
e'c021
e
/* line 55 tst6nopt */
/* E */

5/oc
e'c023
f
/* line 55 tst6nopt */
/* F */

6/oc
e'c029
g
/* line 55 tst6nopt */
/* G */

7/oc
e'c02d
i
/* line 57 tst6nopt */
/* I */

8/oc
e'c02f
k
/* line 57 tst6nopt */
/* K */

9/oc
e'c031
l
/* line 57 tst6nopt */
/* L */

10/oc
e'c039
m
/* line 57 tst6nopt */
/* M */

11/oc
e'c03d
o
/* line 59 tst6nopt */
/* O */

12/oc
e'c03f
p
/* line 59 tst6nopt */
13/oc
e'c043
q
/* line 59 tst6nopt */
/* Q */

14/oc
e'c049
b
/* line 61 tst6nopt */
/* B */

15/oc
e'c04b
d
/* line 61 tst6nopt */
/* D */

16/oc
e'c051
h
/* line 61 tst6nopt */
/* H */

17/oc
e'c057
n
/* line 61 tst6nopt */
/* N */

18/oc
e'c062
numi
/* line 64 tst6nopt */
/* num (i) */

19/oc
e'c063
numi+1
numi1
/* line 64 tst6nopt */
/* num (i+1) */

20/oc
e'c065
numi2
/* line 64 tst6nopt */
/* num *(i+2) */

21/oc
e'c069
loop
/* line 63 testnopt */
/* loop counter i */
@
APPENDIX E

DRS INPUT FILES

DATA RANGE INPUT FILE

\NOVPCMSG\n\NUMFLAGS = 256\n
FORMAT F1 ("Maximum Value" : ", I6");
FORMAT F2 ("Minimum Value" : ", I6");
FORMAT F3 ("Zero Value Did Occur" : ", I6");
FORMAT F4 ("Positive Near Zero Value" : ", I6");
FORMAT F5 ("Negative Near Zero Value" : ", I6");
FORMAT F6 ("RESULTS FOR AM: ", I6");
FORMAT F7 (" ");

MAP
MAXVAL (255) IS "Maximum Value",
MINVAL (255) IS "Minimum Value",
MINPOS (255) IS "Positive Near Zero Value",
MINNEG (255) IS "Negative Near Zero Value",  

97
ZERO (255) IS "Zero Value Did Occur" ;

COUNTER
   AMCOUNT ;
/* AM EVENT */
/* TRYING TO TAKE CARE OF THE PROBLEM OF GETTING OPERAND*/
/* CAPTURE WITHOUT AN AM EVENT PRECEEDING IT */ When evtype eql 10 ,
   increment amcount by 1;

/* OPERAND CAPTURE EVENT = 11 */
When (evtype eql 11) and (val(amcount) eql 1),
   decrement amcount by 1,
/* SET AN INDICATION OF WHICH AM MATCH HAS OCCURRED */
   set flag (ocavail) = 1,
/* TEST FOR MAX VAL */
If freq (maxval, ocavail) eql 0 then
/* SET INITIAL VALUE FOR THAT OPERAND */
MAP OCAVAIL INTO MAXVAL BY DATA,
ELSE
/* OCAVAIL IS WHICH AM YOU ARE DEALING WITH, I.E. 1,2,...ETC */
IF FREQ(MAXVAL,OCAVAIL) LES DATA THEN
/* ZERO THE OLD MAP VALUE OUT */
MAP OCAVAIL INTO MAXVAL BY - FREQ (MAXVAL,OCAVAIL),
/* PUT NEW DATA VALUE IN */
MAP OCAVAIL INTO MAXVAL BY DATA
ENDIF,
ENDIF,

/* TEST FOR MIN VAL */
IF FREQ (MINVAL, OCAVAIL) EQL 0 THEN
MAP OCAVAIL INTO MINVAL BY DATA,
ELSE
IF FREQ(MINVAL,OCAVAIL) GRT DATA THEN
MAP OCAVAIL INTO MINVAL BY - FREQ (MINVAL,OCAVAIL),
MAP OCAVAIL INTO MINVAL BY DATA
ENDIF,
ENDIF,

IF DATA EQL 0 THEN
MAP OCAVAIL INTO ZERO BY -1
ELSE IF DATA GRT 0 THEN
/* MAP MINIMUM VALUE IN */
IF FREQ (MINPOS,OCAVAIL) EQL 0 THEN
MAP OCAVAIL INTO MINPOS BY DATA,
ELSE
IF FREQ (MINPOS,OCAVAIL) GRT DATA THEN
MAP OCAVAIL INTO MINPOS BY - FREQ (MINPOS,OCAVAIL),
MAP OCAVAIL INTO MINPOS BY DATA
ENDIF,
ENDIF,
ELSE /* CHECK FOR MINIMUM NEGATIVE VALUE */
IF FREQ (MINNEG, OCAVAIL) EQL 0 THEN
MAP OCAVAIL INTO MINNEG BY DATA,
ELSE
  IF FREQ (MINNEG, OCAVAIL) LES DATA THEN
    MAP OCAVAIL INTO MINNEG BY - FREQ (MINNEG, OCAVAIL),
    MAP OCAVAIL INTO MINNEG BY DATA
  ENDIF,
  ENDIF,
ENDIF;

FINALLY,
IF FLAG(0) EQL 1 THEN
  LIST F7,
  LIST F6 ( 0 ),
  LIST F7,
  LIST F1 ( FIX (FREQ(MAXVAL, 0)) ),
  LIST F2 ( FIX (FREQ(MINVAL, 0)) ),
  IF FREQ(ZERO, 0) EQL -1 THEN
    LIST F3
ENDIF,
IF FREQ(MINPOS, 0) GRT 0 THEN
   LIST F4 (FIX (FREQ(MINPOS, 0)))
ENDIF,
IF FREQ(MINNEG, 0) LES 0 THEN
   LIST F5 (FIX (FREQ(MINNEG, 0)))
ENDIF,
ENDIF,
IF FLAG(1) EQL 1 THEN
   LIST F7,
   LIST F6 (FIX (1)),
   LIST F7,
   LIST F1 (FIX (FREQ(MAXVAL, 1))),
   LIST F2 (FIX (FREQ(MINVAL, 1))),
   IF FREQ(ZERO, 1) EQL -1 THEN
      LIST F3
   ENDIF,
   IF FREQ(MINPOS, 1) GRT 0 THEN
      LIST F4 (FIX (FREQ(MINPOS, 1)))
   ENDIF,
   IF FREQ(MINNEG, 1) LES 0 THEN
      LIST F5 (FIX (FREQ(MINNEG, 1)))
   ENDIF,
ENDIF,
ENDIF,
IF FLAG(2) EQL 1 THEN
   LIST F7,
   LIST F6 (FIX(2)),
   LIST F7,
   LIST F1 (FIX (FREQ(MAXVAL, 2))),
   LIST F2 (FIX (FREQ(MINVAL, 2))),
   IF FREQ(ZERO, 2) EQL -1 THEN
      LIST F3
   ENDIF,
   IF FREQ(MINPOS, 2) GRT 0 THEN
      LIST F4 (FIX (FREQ(MINPOS, 2)))
   ENDIF,
   IF FREQ(MINNEG, 2) LES 0 THEN
LIST F5 (FIX (FREQ(MINNEG, 2)))
ENDIF,
ENDIF,
IF FLAG(3) EQL 1 THEN
LIST F7,
LIST F6 ( FIX (3) ),
LIST F7 ,
LIST F1 (FIX (FREQ(MAXVAL, 3))),
LIST F2 (FIX (FREQ(MINVAL, 3))),
IF FREQ(ZERO,3) EQL -1 THEN
LIST F3
ENDIF,
IF FREQ(MINPOS, 3) GRT 0 THEN
LIST F4 (FIX (FREQ(MINPOS, 3)))
ENDIF,
IF FREQ(MINNEG, 3) LES 0 THEN
LIST F5 (FIX (FREQ(MINNEG, 3)))
DRS INPUT FILES

ENDIF,
ENDIF,

IF FLAG(4) EQL 1 THEN
  LIST F7,
  LIST F6 ( FIX(4) ),
  LIST F7,
  LIST F1 ( FIX ( FREQ(MAXVAL, 4) ) ),
  LIST F2 ( FIX ( FREQ(MINVAL, 4) ) ),
  IF FREQ(ZERO, 4) EQL -1 THEN
    LIST F3
  ENDIF,

  IF FREQ(MINPOS, 4) GRT 0 THEN
    LIST F4 ( FIX ( FREQ(MINPOS, 4) ) )
  ENDIF,

  IF FREQ(MINNEG, 4) LES 0 THEN
    LIST F5 ( FIX ( FREQ(MINNEG, 4) ) )
  ENDIF,
ENDIF,

IF FLAG(5) EQL 1 THEN
  LIST F7,
  LIST F6 ( FIX(5) ),
  LIST F7,
  LIST F1 ( FIX ( FREQ(MAXVAL, 5) ) ),
  LIST F2 ( FIX ( FREQ(MINVAL, 5) ) ),
  IF FREQ(ZERO,5) EQL -1 THEN
    LIST F3
  ENDIF,

  IF FREQ(MINPOS, 5) GRT 0 THEN
    LIST F4 ( FIX ( FREQ(MINPOS, 5) ) )
  ENDIF,

  IF FREQ(MINNEG, 5) LES 0 THEN
    LIST F5 ( FIX ( FREQ(MINNEG, 5) ) )
  ENDIF,
ENDIF;
PRINT FULL SUMMARY
    USING MAXVAL, MINVAL, MINPOS, MINNEG, AMMAP+;
COMBINATION TRUTH TABLE REPORT INPUT FILE

\NOVPCMSG\\LAST=6000\FORMAT F1 (" TRUTH TABLE ");\FORMAT F2 (" (A OR C) AND (X AND Y)");\FORMAT F3 (S'2',I'1',S'9',I'1',S'8',I'1',S'7',I'1');\FORMAT F4 (" ");\FORMAT F5 (" E AND F OR G ");\FORMAT F6 (" I OR K AND L OR M ");\FORMAT F7 (" (0 OR P) AND Q ");\FORMAT F8 (" B AND D AND H AND N ");\FORMAT F9 ("NUM(I) ANDB NUM(I+1) ORB NUM(I+2) ");\FORMAT F10 (S'2',I'1',S'13',I'1',S'15',I'1');\FORMAT F11 ("LOOP COUNTER I = ", I3');

MAP
AMOUNT(255) IS "AM counter ",
TABLE (255) IS "Boolean values",
VALUE1 (255) IS "first value ",
VALUE2 (255) IS "second value ",
VALUE3 (255) IS "third value ",
VALUE4 (255) IS "fourth value ",
VALUE5 (255) IS "fifth value ",
VALUE6 (255) IS ""
VALUE7 (255) IS ""
VALUE8 (255) IS ""
VALUE9 (255) IS "";

/* OPERAND CAPTURE EVENT = 11 */
WHEN EVTTYPE EQL 11,
IF DATA EQL 0 THEN
MAP OCAVAIL INTO TABLE BY DATA
ELSE
MAP OCAVAIL INTO TABLE BY -FREQ(TABLE,OCAVAIL),
MAP OCAVAIL INTO TABLE BY 1
ENDIF,

MAP OCAVAIL INTO AMCOUNT BY 1,

/* ALLOWS UP TO 4 OCCURANCES OF THE SAME OPERAND CAPTURE */
IF FREQ(AMCOUNT, OCAVAIL) EQL 1 THEN
  MAP OCAVAIL INTO VALUE1 BY FREQ(TABLE, OCAVAIL)
ELSEIF FREQ(AMCOUNT, OCAVAIL) EQL 2 THEN
  MAP OCAVAIL INTO VALUE2 BY FREQ(TABLE, OCAVAIL)
ELSEIF FREQ(AMCOUNT, OCAVAIL) EQL 3 THEN
  MAP OCAVAIL INTO VALUE3 BY FREQ(TABLE, OCAVAIL)
ELSEIF FREQ(AMCOUNT, OCAVAIL) EQL 4 THEN
  MAP OCAVAIL INTO VALUE4 BY FREQ(TABLE, OCAVAIL)
ELSEIF FREQ(AMCOUNT, OCAVAIL) EQL 5 THEN
  MAP OCAVAIL INTO VALUE5 BY FREQ(TABLE, OCAVAIL)
ELSEIF FREQ(AMCOUNT, OCAVAIL) EQL 6 THEN
MAP OCAVAIL INTO VALUE6 BY FREQ(TABLE, OCAVAIL)
ELSEIF FREQ (AMCOUNT, OCAVAIL) EQL 7 THEN
  MAP OCAVAIL INTO VALUE7 BY FREQ(TABLE, OCAVAIL)
ENDIF;

FINALLY,
  LIST F4,
  LIST F1,
  LIST F2,
/* (A OR C) AND (X AND Y) */
  IF FREQ(AMCOUNT,0) GEQ 1 THEN
    LIST F3 (FIX (FREQ(VALUE1,0)),
             FIX (FREQ(VALUE1,1)),
             FIX (FREQ(VALUE1,2)),
             FIX (FREQ(VALUE1,3)))
  ENDIF,
  IF FREQ(AMCOUNT,0) GEQ 2 THEN
    LIST F3 (FIX (FREQ(VALUE2,0)),
             FIX (FREQ(VALUE2,1)),
             FIX (FREQ(VALUE2,2)),
             FIX (FREQ(VALUE2,3)))
  ENDIF,
  IF FREQ(AMCOUNT,0) GEQ 3 THEN
    LIST F3 (FIX (FREQ(VALUE3,0)),
             FIX (FREQ(VALUE3,1)),
             FIX (FREQ(VALUE3,2)),
             FIX (FREQ(VALUE3,3)))
  ENDIF,
  IF FREQ(AMCOUNT,0) GEQ 4 THEN
    LIST F3 (FIX (FREQ(VALUE4,0)),
             FIX (FREQ(VALUE4,1)),
             FIX (FREQ(VALUE4,2)),
             FIX (FREQ(VALUE4,3)))
  ENDIF,
/* E AND F OR G */
  LIST F4,
LIST F1,
LIST F5,
IF FREQ(AMCOUNT, 4) GEQ 1 THEN
  LIST F3 (FIX (FREQ(VALUE1, 4)),
          FIX (FREQ(VALUE1, 5)),
          FIX (FREQ(VALUE1, 6)))
ENDIF,
IF FREQ(AMCOUNT, 4) GEQ 2 THEN
  LIST F3 (FIX (FREQ(VALUE2, 4)),
          FIX (FREQ(VALUE2, 5)),
          FIX (FREQ(VALUE2, 6)))
ENDIF,
IF FREQ(AMCOUNT, 4) GEQ 3 THEN
  LIST F3 (FIX (FREQ(VALUE3, 4)),
          FIX (FREQ(VALUE3, 5)),
          FIX (FREQ(VALUE3, 6)))
ENDIF,
IF FREQ(AMCOUNT,4) GEQ 4 THEN
LIST F3 (FIX (FREQ(VALUE4,4)),
        FIX (FREQ(VALUE4,5)),
        FIX (FREQ(VALUE4,6)))
ENDIF,

/* I OR K AND L OR M */
LIST F4,
LIST F1,
LIST F6,
IF FREQ(AMCOUNT,7) GEQ 1 THEN
LIST F3 (FIX (FREQ(VALUE1,7)),
        FIX (FREQ(VALUE1,8)),
        FIX (FREQ(VALUE1,9)),
        FIX (FREQ(VALUE1,10)))
ENDIF,
IF FREQ(AMCOUNT,7) GEQ 2 THEN
LIST F3 (FIX (FREQ(VALUE2,7)),
        FIX (FREQ(VALUE2,8)),
        FIX (FREQ(VALUE2,9)),
        FIX (FREQ(VALUE2,10)))
ENDIF,
IF FREQ(AMCOUNT,7) GEQ 3 THEN
LIST F3 (FIX (FREQ(VALUE3,7)),
        FIX (FREQ(VALUE3,8)),
        FIX (FREQ(VALUE3,9)),
        FIX (FREQ(VALUE3,10)))
ENDIF,
IF FREQ(AMCOUNT,7) GEQ 4 THEN
LIST F3 (FIX (FREQ(VALUE4,7)),
        FIX (FREQ(VALUE4,8)),
        FIX (FREQ(VALUE4,9)),
        FIX (FREQ(VALUE4,10)))
ENDIF,

/* (O OR P) AND Q */
LIST F4,
LIST F1,
LIST F7,
IF FREQ(AMCOUNT,11) GEQ 1 THEN
   LIST F3 (FIX (FREQ(VALUE1,11)),
            FIX (FREQ(VALUE1,12)),
            FIX (FREQ(VALUE1,13)))
ENDIF,
IF FREQ(AMCOUNT,11) GEQ 2 THEN
   LIST F3 (FIX (FREQ(VALUE2,11)),
            FIX (FREQ(VALUE2,12)),
            FIX (FREQ(VALUE2,13)))
ENDIF,
IF FREQ(AMCOUNT,11) GEQ 3 THEN
   LIST F3 (FIX (FREQ(VALUE3,11)),
            FIX (FREQ(VALUE3,12)),
            FIX (FREQ(VALUE3,13)))
ENDIF,
IF FREQ(AMCOUNT,11) GEQ 4 THEN
    LIST F3 (FIX (FREQ(VALUE4,11)),
    FIX (FREQ(VALUE4,12)),
    FIX (FREQ(VALUE4,13)))
ENDIF,

/* B AND D AND H AND N */
LIST F4,
LIST F1,
LIST F8,
IF FREQ(AMCOUNT,14) GEQ 1 THEN
    LIST F3 (FIX (FREQ(VALUE1,14)),
    FIX (FREQ(VALUE1,15)),
    FIX (FREQ(VALUE1,16)),
    FIX (FREQ(VALUE1,17)))
ENDIF,
IF FREQ(AMCOUNT,14) GEQ 2 THEN
    LIST F3 (FIX (FREQ(VALUE2,14)),
    FIX (FREQ(VALUE2,15)),
    FIX (FREQ(VALUE2,16)),
    FIX (FREQ(VALUE2,17)))
ENDIF,
IF FREQ(AMCOUNT,14) GEQ 3 THEN
    LIST F3 (FIX (FREQ(VALUE3,14)),
    FIX (FREQ(VALUE3,15)),
    FIX (FREQ(VALUE3,16)),
    FIX (FREQ(VALUE3,17)))
ENDIF,
IF FREQ(AMCOUNT,14) GEQ 4 THEN
    LIST F3 (FIX (FREQ(VALUE4,14)),
    FIX (FREQ(VALUE4,15)),
    FIX (FREQ(VALUE4,16)),
    FIX (FREQ(VALUE4,17)))
ENDIF,

/* LOOP COUNTER I */
IF FREQ(AMCOUNT,21) GRT 0 THEN
LIST F4,
LIST F4,
LIST F11 (FIX(FREQ(VALUE1,21))),
LIST F4
ENDIF,

IF FREQ(AMCOUNT,21) GRT 1 THEN
LIST F4,
LIST F4,
LIST F11 (FIX(FREQ(VALUE2,21))),
LIST F4
ENDIF,

IF FREQ(AMCOUNT,21) GRT 2 THEN
LIST F4,
LIST F4,
LIST F11 (FIX(FREQ(VALUE3,21))),
LIST F4
ENDIF,

IF FREQ(AMCOUNT,21) GRT 3 THEN
  LIST F4,
  LIST F4,
  LIST F11 (FIX(FREQ(VALUE4,21))),
  LIST F4
ENDIF,

IF FREQ(AMCOUNT,21) GRT 4 THEN
  LIST F4,
  LIST F4,
  LIST F11 (FIX(FREQ(VALUE5,21))),
  LIST F4
ENDIF,

IF FREQ(AMCOUNT,21) GRT 5 THEN
  LIST F4,
  LIST F4,
  LIST F11 (FIX(FREQ(VALUE6,21))),
  LIST F4
ENDIF,

IF FREQ(AMCOUNT,21) GRT 6 THEN
  LIST F4,
  LIST F4,
  LIST F11 (FIX(FREQ(VALUE7,21))),
  LIST F4
ENDIF,

/* NUM(I) ANDB NUM(I+1) ORB NUM(I+2) */
  LIST F4,
  LIST F1,
  LIST F9,
  IF FREQ(AMCOUNT,18) GEQ 1 THEN
LIST F10 (FIX (FREQ(VALUE1,18)),
       FIX (FREQ(VALUE1,19)),
       FIX (FREQ(VALUE1,20)))
ENDIF,
IF FREQ(AMCOUNT,18) GEQ 2 THEN
  LIST F10 (FIX (FREQ(VALUE2,18)),
             FIX (FREQ(VALUE2,19)),
             FIX (FREQ(VALUE2,20)))
ENDIF,
IF FREQ(AMCOUNT,18) GEQ 3 THEN
  LIST F10 (FIX (FREQ(VALUE3,18)),
             FIX (FREQ(VALUE3,19)),
             FIX (FREQ(VALUE3,20)))
ENDIF,
IF FREQ(AMCOUNT,18) GEQ 4 THEN
  LIST F10 (FIX (FREQ(VALUE4,18)),
             FIX (FREQ(VALUE4,19)),
             FIX (FREQ(VALUE4,20)))
FIX (FREQ(VALUE4,20)))
ENDIF;

PRINT FULL SUMMARY ;
APPENDIX F

DRS REPORTS

REPORT FOR TEST PROGRAM 1
GBROOKS
8-MAR-1990 12:11:03.32 NSWC Dahlgren, Va.
DRS Rev 6.014 Created 27-OCT-1989 12:55 Page 1

* 1 * \NOVPCMSG\ 
* 2 * \NUMFLAGS = 10\ 
* 3 * 
* 4 * FORMAT F1 ('Maximum Value : "', I'6'); 
* 5 * FORMAT F2 ('Minimum Value : "', I'6'); 
* 6 * FORMAT F3 ('Zero Value Did Occur : "', I'6'); 
* 7 * FORMAT F4 ('Positive Near Zero Value : "', I'6'); 
* 8 * FORMAT F5 ('Negative Near Zero Value : "', I'6'); 
* 9 * FORMAT F6 ('RESULTS FGR AM: ";', I'6'); 
*10 * FORMAT F7 ('") ; 
*11 * 
*12 * MAP 
*13 * MAXVAL (255) IS "Maximum Value", 
*14 * MINVAL (255) IS "Minimum Value", 
*15 * MINPOS (255) IS "Positive Near Zero Value", 
*16 * MINNEG (255) IS "Negative Near Zero Value", 
*17 * ZERO (255) IS "Zero Value Did Occur" ; 
*18 * 
*19 * /* OPERAND CAPTURE EVENT = 11 */ 
*20 * WHEN EVTYP = EQL 11, 
*21 * /* SET AN INDICATION OF WHICH AM MATCH HAS OCCURRED */ 
*22 * SET FLAG (OCAVAL) = 1, 
*23 * /* TEST FOR MAX VAL */ 
*24 * IF FREQ (MAXVAL, OCAVAL) EQL 0 THEN 
*25 * /* SET INITIAL VALUE FOR THAT OPERAND */ 
*26 * MAP OCAVAL INTO MAXVAL BY DATA, 
*27 * ELSE 
*28 * /* OCAVAL IS WHICH AM YOU ARE DEALING WITH, I.E. 
1, 2, ... ETC */ 
*29 * IF FREQ (MAXVAL, OCAVAL) LES DATA THEN 
*30 * /* ZERO THE OLD MAP VALUE OUT */ 
*31 * MAP OCAVAL INTO MAXVAL BY -FLAG (OCAVAL), 
*32 * /* PUT NEW DATA VALUE IN */ 
*33 * MAP OCAVAL INTO MAXVAL BY DATA 
*34 * ENDIF , 
*35 * ENDIF ,
* 36 *
* 37 * /* TEST FOR MIN VAL */

DRS REPORTS

* 38 * IF FREQL(MINVAL, OCAVAIL) EQL 0 THEN
* 39 * MAP OCAVAIL INTO MINVAL BY DATA,
* 40 * ELSE
* 41 * IF FREQL(MINVAL, OCAVAIL) GRT DATA THEN
* 42 * MAP OCAVAIL INTO MINVAL BY -FLAG(OCAVAIL),
* 43 * MAP OCAVAIL INTO MINVAL BY DATA
* 44 * ENDDIF,
* 45 * ENDDIF,
* 46 *
* 47 * IF DATA EQL 0 THEN
* 48 * MAP -1 INTO ZERO
* 49 * ELSEIF DATA GRT 0 THEN
* 50 * /* MAP MINIMUM VALUE IN */
* 51 * IF FREQL(MINPOS, OCAVAIL) EQL 0 THEN
* 52 * MAP OCAVAIL INTO MINPOS BY DATA,
* 53 * ELSE
* 54 * IF FREQL(MINPOS, OCAVAIL) GRT DATA THEN
* 55 * MAP OCAVAIL INTO MINPOS BY -FLAG(OCAVAIL),
* 56 * MAP OCAVAIL INTO MINPOS BY DATA

GBROOKS

8-MAR-1990 12:11:03.32 NSWC Dahlgren, Va.

* 57 *
* 58 * ENDDIF,
* 59 * ELSE /* CHECK FOR MINIMUM NEGATIVE VALUE */
* 60 * IF FREQL(MINNEG, OCAVAIL) EQL 0 THEN
* 61 * MAP OCAVAIL INTO MINNEG BY DATA,
* 62 * ELSE
* 63 * IF FREQL(MINNEG, OCAVAIL) LES DATA THEN
* 64 * MAP OCAVAIL INTO MINNEG BY -FLAG(OCAVAIL),
* 65 * MAP OCAVAIL INTO MINNEG BY DATA
* 66 * ENDDIF,
* 67 * ENDDIF,
* 68 * ENDDIF;
* 69 *
* 70 * FINALLY,
* 71 * IF FLAG(0) EQL 1 THEN
* 72 * LIST F7,
* 73 * LIST F6 ( 0 ),
* 74 * LIST F7,
* 75 * LIST F1 ( FIX(FREQL(MAXVAL, 0)))
* 76 * LIST F2 ( FIX(FREQL(MINVAL, 0)))
* 77 * IF FREQL(ZERO, 0) EQL -1 THEN
* 78 * LIST F3
* 79 * ENDDIF,
* 80 * IF FREQL(MINPOS, 0) GRT 0 THEN
* 81 * LIST F4 ( FIX(FREQL(MINPOS, 0)))
* 82 * ENDDIF,
* 83 * IF FREQL(MINNEG, 0) LES 0 THEN
* 84 * LIST F5 ( FIX(FREQL(MINNEG, 0)))
* 85 * ENDDIF,
* 86 * ENDIF,
* 87 * IF FLAG(1) EQL 1 THEN
* 88 * LIST F6 ( 1 ),

DRS REPORTS

* 89 * LIST F1 (FIX(FREQ(MAXVAL, 1))),
* 90 * LIST F2 (FIX(FREQ(MINVAL, 1))),
* 91 * IF FREQ(ZERO, 1) EQL -1 THEN
* 92 * LIST F3
* 93 * ENDIF,
* 94 * LIST F4 (FIX(FREQ(MINPOS, 1))),
* 95 * LIST F5 (FIX(FREQ(MINNEG, 1))),
* 96 * ENDIF;
* 97 *
* 98 * PRINT FULL SUMMARY
* 99 * USING MAXVAL, MINVAL, MINPOS, MINNEG, AMMAP$$;

% DRS - I, End of File
UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]TEST1.DRS;

GBROOKS 8-MAR-1990 12:11:03.32 NSWC Dahlgren, Va.

SYMBOL REFERENCE

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>SYM</th>
<th>NODE</th>
<th>SYM</th>
<th>ETS-STB</th>
<th>ETS</th>
<th>ASSOCIATED TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAPPING</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRING</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>ADDR</td>
<td>TYPE</td>
<td>ADDR</td>
<td>LENGTH</td>
<td></td>
<td>(IF APPLICABLE)</td>
</tr>
<tr>
<td>LINE</td>
<td>FILE</td>
<td>SPECIFICATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

******************************************************************************

******************************************************************************

Symbols in Block 1. Starting at Line 1 and Ending at Line 0
AMMAP$$  C2448  MAP  BC3A8  300  Associative Memory
Mapping  2  TEST1.PDV
EVTMAP$$  C2470  MAP  BC278  76  Event Type Code
Distribution  1  TEST1.PDV
EXEC  C23D0  SYNON  C23A8  1
      5  TEST1.PDV  FMT  BC858  15
      4  TEST1.DRS  FMT  BC894  15
F1  C2308  FMT  BC8D0  15
F2  C22E0  FMT  BC90C  15
F3  C22B8  FMT  BC948  15
F4  C2290  FMT  BC984  12
F5  C2268  FMT  BC9B4  8
F6  C2240  FMT  TEST1.DRS
      9  TEST1.DRS
F7  C2218  MAP  BC9D4  300  Maximum Value
      13  TEST1.DRS
MAXVAL  C21A0  MAP  BD7E4  300  Negative Near Zero
Value  16  TEST1.DRS
MINPOS  C21C8  MAP  BD334  300  Positive Near Zero
Value 15 TEST1.DRS
MINVAL C21F0 MAP BCE84 300 Minimum Value
14 TEST1.DRS
OPERANDX C24A8 PDV E C2498 1 AM ( 1)
0 TEST1.PDV
OPERANDY C2480 PDV E C24D0 1 AM ( 0)
0 TEST1.PDV
TASK C2420 SYNON C23F8 1
4 TEST1.PDV
TBP_ID C2380 SYNON C2358 1
6 TEST1.PDV
ZERO C2178 MAP BDC94 300 Zero Value Did Occur
17 TEST1.DRS

******************************************************************************
******************************************************************************
Synonym Stack Address C0288 Length 64 Used 0
Item Encoding Stack Address C0338 Length 512 Used 25
Symbol Table Address C0B82 Length 3048 Used 1646
Event Tracking Storage Address BBFD0 Length 4096 Used 2627

Input Files Read
( 0) UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]TEST1.DRS;
( 1) NSWC$ROOT:[INC]TADPOLE.DRS.EXTEND;
( 2) UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]TEST1.PDV;

DRS REPORTS

EVENT OCCURRENCE TRACE
Page 4
7-MAR-1990 10:28:36.50
Page F-4

Session TEST1
TFCS Mode Software Version
Sequence 1

ELAPSED TIME EVENT OCCURRENCE STATE
TID MID TS ASSOCIATED DATA

RESULTS FOR AM: 0
Maximum Value : 1
Minimum Value : 1
Positive Near Zero Value : 1

RESULTS FOR AM: 1
Maximum Value : 1
Minimum Value : 1
Positive Near Zero Value : 1

Execution Summary (CPU Seconds)
System Initialization 0.013
<table>
<thead>
<tr>
<th>User Interface</th>
<th>0.095</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDV Processing</td>
<td>0.014</td>
</tr>
<tr>
<td>Report Generation</td>
<td>0.012</td>
</tr>
<tr>
<td>Program Coverage</td>
<td>0.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Performance Summary Length</th>
<th>4229 Words</th>
<th>34 PRUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPG Command File Length</td>
<td>51 Words</td>
<td>1 PRUs</td>
</tr>
<tr>
<td>Buffered I-O Count</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Direct I-O Count</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>Number of Page Faults</td>
<td>391</td>
<td></td>
</tr>
<tr>
<td>Dynamic Memory Utilization</td>
<td>51712 Words</td>
<td></td>
</tr>
<tr>
<td>Turn-around Time</td>
<td>00:00:03.48</td>
<td></td>
</tr>
</tbody>
</table>

Fatal Errors Were Not Encountered

Command Option Summary
/INPUT=UDISK4: [GBROOKS.CLASS.SPRING90.ENG5904.TEST]TEST1.DRS;
/OUTPUT=UDISK4: [GBROOKS.CLASS.SPRING90.ENG5904.TEST]TEST1.RPT;
/PDV=UDISK4: [GBROOKS.CLASS.SPRING90.ENG5904.TEST]TEST1.PDV;
/MSGFIL=NSWC$ROOT: [DAT]TADPOLE_DRS.DMF
/PSFOUT=UDISK4: [GBROOKS.CLASS.SPRING90.ENG5904.TEST]TEST1.PSF; Was NOT Saved
/RPGCF=DRSRPGCF.DAT
/PILOT=UDISK4: [GBROOKS.CLASS.SPRING90.ENG5904.TEST]TEST1.CCDRAW;
/NCPSFIN
/NOSCHEMA
/EXTEND=NSWC$ROOT: [INC]TADPOLE_DRS.EXTEND;
/NGDVP
/SYM
/LIST
DRS REPORTS
REPORT FOR TEST PROGRAM 2

GBROOKS
DRS Rev 6.014 Created 27-OCT-1989 12:55 Page 1

* 1 * \NOPCMSG\n* 2 * \NUMFLAGS = 256\n* 3 *
* 4 * FORMAT F1 ("Maximum Value : ", I'6');
* 5 * FORMAT F2 ("Minimum Value : ", I'6');
* 6 * FORMAT F3 ("Zero Value Did Occur : ", I'6');
* 7 * FORMAT F4 ("Positive Near Zero Value : ", I'6');
* 8 * FORMAT F5 ("Negative Near Zero Value : ", I'6');
* 9 * FORMAT F6 ("RESULTS FOR AM: ", I'6');
* 10 * FORMAT F7 (" ");
* 11 *
* 12 * MAP
* 13 * MAXVAL (255) IS "Maximum Value",
* 14 * MINVAL (255) IS "Minimum Value",
* 15 * MINPOS (255) IS "Positive Near Zero Value",
* 16 * MINNEG (255) IS "Negative Near Zero Value",
* 17 * ZERO (255) IS "Zero Value Did Occur" ;
* 18 *
* 19 * /* OPERAND Capture Event = 11 */
* 20 * WHEN EVTYP = 11,
* 21 * /* SET AN INDICATION OF WHICH AM MATCH HAS OCCURRED */
* 22 * SET FLAG (OCAVAIL) = 1,
* 23 * /* TEST FOR MAX VAL */
* 24 * IF FREQ (MAXVAL, OCAVAIL) EQ 0 THEN
* 25 * /* SET INITIAl VALUE FOR THAT OPERAND */
* 26 * MAP OCAVAIL INTO MAXVAL BY DATA,
* 27 * ELSE /* OCAVAIL IS WHICH AM YOU ARE DEALING WITH, I.E.
* 28 * 1,2,...ETC */
* 29 * IF FREQ(MAXVAL,OCAVAIL) LES DATA THEN
* 30 * /* ZERO THE OLD MAP VALUE OUT */
* 31 * MAP OCAVAIL INTO MAXVAL BY - FREQ (MAXVAL,OCAVAIL),
* 32 * /* PUT NEW DATA VALUE IN */
* 33 * MAP OCAVAIL INTO MAXVAL BY DATA
* 34 * ENDF,
* 35 * ENDF,
* 36 *
* 37 * /* TEST FOR MIN VAL */
* 38 * IF FREQ (MINVAL, OCAVAIL) EQ 0 THEN
* 39 * MAP OCAVAIL INTO MINVAL BY DATA,
* 40 * ELSE
* 41 * IF FREQ(MINVAL,OCAVAIL) GRT DATA THEN
* 42 * MAP OCAVAIL INTO MINVAL BY - FREQ (MINVAL,OCAVAIL),
* 43 * MAP OCAVAIL INTO MINVAL BY DATA
* 44 * ENDF,
* 45 * ENDF,
* 46 *
* 47 * IF DATA EQ 0 THEN
* 48 * MAP -1 INTO ZERO
* 49 * ELSEIF DATA GRT 0 THEN
* 50 * /* MAP MINIMUM VALUE IN */
* 51 * IF FREQ (MINPOS,OCAVAIL) EQ 0 THEN

DRS REPORTS
MAP OCAVAIL INTO MINPOS BY DATA,
ELSE
IF FREQ(MINPOS,OCAVAIL) GRT DATA THEN
MAP OCAVAIL INTO MINPOS BY - FREQ(MINPOS,OCAVAIL),
MAP OCAVAIL INTO MINPOS BY DATA
ENDIF,
ELSE
IF FREQ(MINNEG,OCAVAIL) LES DATA THEN
MAP OCAVAIL INTO MINNEG BY - FREQ(MINNEG,OCAVAIL),
MAP OCAVAIL INTO MINNEG BY DATA
ENDIF,
ENDIF;
FINALLY,
IF FLAG(0) EQL 1 THEN
LIST F7,
LIST F6 ( 0 ),
LIST F7,
LIST F1 (FIX(FREQ(MAXVAL, 0))),
LIST F2 (FIX(FREQ(MINVAL, 0))),
IF FREQ(ZERO, 0) EQL -1 THEN
LIST F3
ENDIF,
IF FREQ(MINPOS, 0) GRT 0 THEN
LIST F4 (FIX(FREQ(MINPOS, 0)))
ENDIF,
IF FREQ(MINNEG, 0) LES 0 THEN
LIST F5 (FIX(FREQ(MINNEG, 0)))
ENDIF,
FINALLY,
IF FLAG(1) EQL 1 THEN
LIST F6 ( 1 ),
LIST F1 (FIX(FREQ(MAXVAL, 1))),
LIST F2 (FIX(FREQ(MINVAL, 1))),
IF FREQ(ZERO,1) EQL -1 THEN
LIST F3
ENDIF,
IF FREQ(MINPOS, 1) GRT 0 THEN
LIST F4 (FIX(FREQ(MINPOS, 1)))
ENDIF,
IF FREQ(MINNEG, 1) LES 0 THEN
LIST F5 (FIX(FREQ(MINNEG, 1)))
ENDIF;
PRINT FULL SUMMARY
** 103 **  USING MAXVAL, MINVAL, MINPOS, MINNEG, AMMAP$$;
%DRS-I, End of File DUSK4:[GBROOKS.CLASS.SPRING90.ENG5904.TEST]DRS.IN;


**SYMBOL REFERENCE**

**MAPPING**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>NODE</th>
<th>ETS-STE</th>
<th>ETS</th>
<th>ASSOCIATED TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>NAME</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>ADDR</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>ADDR</strong></td>
</tr>
</tbody>
</table>

**END OF SYMBOL REFERENCE**

**END OF MAPPING**

**SYMBOLS in Block 1. Starting at Line 1 and Ending at Line 0**

**AMMAP$$**
- C2478 MAP BCB50 300 Associative Memory

**Mapping 2**
- EVTMAP$$
  - C24A0 MAP BCA20 76 Event Type Code

**Distribution**
- EXEC C2400 SYNON C23D8 1
  - F1 C2360 FMT BD000 15
  - F2 C2338 FMT BD03C 15
  - F3 C2310 FMT BD078 15
  - F4 C22E8 FMT BD0B4 15
  - F5 C22C0 FMT BD0F0 15
  - F6 C2298 FMT BD12C 12
  - F7 C2270 FMT BD15C 8
  - LOADI C24D8 PDV E C24C8 1 AM( 0)
- MAXVAL C2248 MAP BD17C 300 Maximum Value
- MINNEG C21D0 MAP BDF8C 300 Negative Near Zero
- MINPOS C21F8 MAP BDADC 300 Positive Near Zero
- MINVAL C2220 MAP BD62C 300 Minimum Value
- TASK C2450 SYNON C2428 1
- TBP_ID C23B0 SYNON C2388 1
- ZERO C21A8 MAP BE43C 300 Zero Value Did Occur

**END OF SYMBOL REFERENCE**

**END OF MAPPING**

**END OF SYMBOLS in Block 1. Starting at Line 1 and Ending at Line 0**

**END OF END OF SYMBOLS**

### Synonym Stack Address
- C0280 Length 64 Used 0

### Item Encoding Stack Address
- C0380 Length 512 Used 25
Symbol Table Address COB80 Length 3048 Used 1632
Event Tracking Storage Address BBFC8 Length 4096 Used 3159

Input Files Read
(0) UDISK4:[GROOKS.CLASS.SPRING90.ENGR5904.TEST]DRS.IN;
(1) N8WC$ROOT:[INC]TADPOLE.DRS.EXTEND;
(2) UDISK4:[GROOKS.CLASS.SPRING90.ENGR5904.TEST]TEST2.PDV;

EVENT OCCURRENCE TRACE
Page 4

7-MAR-1990 10:59:29.50

Session TEST2 Sequence 1
TFC8 Mode Software Version

ELAPSED TIME EVENT OCCURRENCE STATE PID
TID MID TS ASSOCIATED DATA

RESULTS FOR AM:

Maximum Value : 24
Minimum Value : 0
Zero Value Did Occur :

DRS REPORTS

Positive Near Zero Value : 1

RESULTS FOR AM:

Maximum Value : 23
Minimum Value : 0
Zero Value Did Occur :
Positive Near Zero Value : 1

Execution Summary (CPU Seconds)

System Initialization 0.009
User Interface 0.066
PDV Processing 0.043
Report Generation 0.009
Program Coverage 0.000

Performance Summary Length 8057 Words 63 PRUs
RPG Command File Length 51 Words 1 PRUs
Buffered I-O Count 20
Direct I-O Count 47
Number of Page Faults 442
Dynamic Memory Utilization 84608 Words
Turn-around Time 00:00:03.62

Fatal Errors Were Not Encountered
Command Option Summary

/INPUT=UDISK4:[GBROOKS.CLASS.SPRING90.ENG5904.TEST]DRS.IN;
/OUTPUT=UDISK4:[GBROOKS.CLASS.SPRING90.ENG5904.TEST]TEST2.RPT;
/PDV=UDISK4:[GBROOKS.CLASS.SPRING90.ENG5904.TEST]TEST2.PDV;
/MSGFIL=NSWC$ROOT:[DAT]TADPOLE_DRS.DMF
/PSFOUT=UDISK4:[GBROOKS.CLASS.SPRING90.ENG5904.TEST]TEST2.PSF; Was Saved
/RPGCF=DRSRPGCF.DAT
/PLOT=UDISK4:[GBROOKS.CLASS.SPRING90.ENG5904.TEST]DRS.CCDRAW;
/NOPSFIN
/NOSCHEM
/EXTEND=NSWC$ROOT:[INC]TADPOLE_DRS.EXTEND;
/NODDP
/SYM
/LIST

DRS REPORTS
REPORT FOR TEST PROGRAM 3

GBROOKS

DRS Rev 6.014 Created 27-OCT-1989 12:55 Page 1

* 1 * \NOVPCMSG\ 
* 2 * \NUMFLAGS = 256\ 
* 3 * 
* 4 * FORMAT F1 ("Maximum Value : ", I'6'); 
* 5 * FORMAT F2 ("Minimum Value : ", I'6'); 
* 6 * FORMAT F3 ("Zero Value Did Occur : ", I'6'); 
* 7 * FORMAT F4 ("Positive Near Zero Value : ", I'6'); 
* 8 * FORMAT F5 ("Negative Near Zero Value : ", I'6'); 
* 9 * FORMAT F6 ("RESULTS FOR AM: ", I'6'); 
* 10 * FORMAT F7 (" "); 
* 11 * 
* 12 * MAP 
* 13 * MAXVAL (255) IS "Maximum Value", 
* 14 * MINVAL (255) IS "Minimum Value", 
* 15 * MINPOS (255) IS "Positive Near Zero Value", 
* 16 * MINNEG (255) IS "Negative Near Zero Value", 
* 17 * ZERO (255) IS "Zero Value Did Occur"; 
* 18 * 
* 19 * COUNTER 
* 20 * AMOUNT; 
* 21 * /* AM EVENT */ 
* 22 * /* TRYING TO TAKE CARE OF THE PROBLEM OF GETTING OPERAND*/ 
* 23 * /* CAPTURE WITHOUT AN AM EVENT PRECEEDING IT */ 
* 24 * WHEN EVTYPE EQL 10, 
* 25 * INCREMENT AMOUNT BY 1; 
* 26 * 
* 27 * /* OPERAND CAPTURE EVENT = 11 */ 
* 28 * WHEN (EVTYPE EQL 11) AND (VAL(AMOUNT) EQL 1), 
* 29 * DECREMENT AMOUNT BY 1, 
* 30 * /* SET AN INDICATION OF WHICH AM MATCH HAS OCCURRED */ 
* 31 * SET FLAG (OCAVAIL) = 1, 
* 32 * /* TEST FOR MAX VAL */ 
* 33 * IF FREQ (MAXVAL, OCAVAIL) EQL 0 THEN 
* 34 * /* SET INITIAL VALUE FOR THAT OPERAND */ 
* 35 * MAP OCAVAIL INTO MAXVAL BY DATA, 
* 36 * ELSE 
* 37 * /* OCAVAIL IS WHICH AM YOU ARE DEALING WITH, I.E. 
1,2,...ETC */ 
* 38 * IF FREQ(MAXVAL,OCAVAIL) LES DATA THEN 
* 39 * /* ZERO THE OLD MAP VALUE OUT */ 
* 40 * MAP OCAVAIL INTO MAXVAL BY - FREQ (MAXVAL,OCAVAIL), 
* 41 * /* PUT NEW DATA VALUE IN */ 
* 42 * MAP OCAVAIL INTO MAXVAL BY DATA 
* 43 * ENDIF, 
* 44 * ENDIF, 
* 45 * 
* 46 * /* TEST FOR MIN VAL */ 
* 47 * IF FREQ (MINVAL, OCAVAIL) EQL 0 THEN 
* 48 * MAP OCAVAIL INTO MINVAL BY DATA, 
* 49 * ELSE 
* 50 * IF FREQ(MINVAL,OCAVAIL) GRT DATA THEN
* 51 * MAP OCAVAIL INTO MINVAL BY - FREQ (MINVAL, OCAVAIL),
* 52 * MAP OCAVAIL INTO MINVAL BY DATA
* 53 * ENDIF,
* 54 * ENDIF,
* 55 * IF DATA EQL 0 THEN
* 56 *

GBROOKS 28-MAR-1990 10:14:05.50 NSWC Dahlgren, Va.

* 57 * MAP OCAVAIL INTO ZERO BY -1
* 58 * ELSEIF DATA GRT 0 THEN
* 59 * /* MAP MINIMUM VALUE IN */
* 60 * IF FREQ (MINPOS, OCAVAIL) EQL 0 THEN
* 61 * MAP OCAVAIL INTO MINPOS BY DATA,
* 62 * ELSE
* 63 * IF FREQ (MINPOS, OCAVAIL) GRT DATA THEN
* 64 * MAP OCAVAIL INTO MINPOS BY - FREQ (MINPOS, OCAVAIL),
* 65 * MAP OCAVAIL INTO MINPOS BY DATA
* 66 * ENDIF,
* 67 * ENDIF,
* 68 * ELSE /* CHECK FOR MINIMUM NEGATIVE VALUE */
* 69 * IF FREQ (MINNEG, OCAVAIL) EQL 0 THEN
* 70 * MAP OCAVAIL INTO MINNEG BY DATA,
* 71 * ELSE
* 72 * IF FREQ (MINNEG, OCAVAIL) LES DATA THEN
* 73 * MAP OCAVAIL INTO MINNEG BY - FREQ (MINNEG, OCAVAIL),
* 74 * MAP OCAVAIL INTO MINNEG BY DATA
* 75 * ENDIF,
* 76 * ENDIF,
* 77 * ENDIF;
* 78 *

* 79 * FINALLY,
* 80 * IF FLAG(0) EQL 1 THEN
* 81 * LIST F7,
* 82 * LIST F6 ( 0 ),
* 83 * LIST F7,
* 84 * LIST F1 ( FIX (FREQ(MAXVAL, 0)) ),
* 85 * LIST F2 ( FIX (FREQ(MINVAL, 0)) ),
* 86 * IF FREQ(ZERO, 0) EQL -1 THEN
* 87 * LIST F3
* 88 * ENDIF,
* 89 * IF FREQ(MINPOS, 0) GRT 0 THEN
* 90 * LIST F4 ( FIX (FREQ(MINPOS, 0)) )
* 91 * ENDIF,
* 92 * IF FREQ(MINNEG, 0) LES 0 THEN
* 93 * LIST F5 ( FIX (FREQ(MINNEG, 0)) )
* 94 * ENDIF,
* 95 * ENDIF,
* 96 * IF FLAG(1) EQL 1 THEN
* 97 * LIST F7,
* 98 * LIST F6 ( FIX (1) ),
* 99 * LIST F7,
* 100 * LIST F1 ( FIX (FREQ(MAXVAL, 1)) ),
* 101 * LIST F2 ( FIX (FREQ(MINVAL, 1)) ),

DRS REPORTS Page F-11
* 102 * IF FREQ(ZERO, 1) EQL -1 THEN
* 103 * LIST F3
* 104 * ENDIF,
* 105 * IF FREQ(MINPOS, 1) GRT 0 THEN
* 106 * LIST F4 (FIX (FREQ(MINPOS, 1)))
* 107 * ENDIF,
* 108 * IF FREQ(MINNEG, 1) LES 0 THEN
* 109 * LIST F5 (FIX (FREQ(MINNEG, 1)))
* 110 * ENDIF,
* 111 * ENDIF,
* 112 * IF FLAG(2) EQL 1 THEN
* 113 * LIST F7,

GBROOKS 28-MAR-1990 10:14:05.50 NSWC Dahlgren, Va.

* 114 * LIST F6 (FIX(2)),
* 115 * LIST F7,
* 116 * LIST F1 (FIX (FREQ(MAXVAL, 2))),
* 117 * LIST F2 (FIX (FREQ(MINVAL, 2))),
* 118 * IF FREQ(ZERO, 2) EQL -1 THEN
* 119 * LIST F3
* 120 * ENDIF,
* 121 * IF FREQ(MINPOS, 2) GRT 0 THEN
* 122 * LIST F4 (FIX (FREQ(MINPOS, 2)))
* 123 * ENDIF,
* 124 * IF FREQ(MINNEG, 2) LES 0 THEN
* 125 * LIST F5 (FIX (FREQ(MINNEG, 2)))
* 126 * ENDIF,
* 127 * ENDIF,
* 128 * IF FLAG(3) EQL 1 THEN
* 129 * LIST F7,
* 130 * LIST F6 (FIX (3)),
* 131 * LIST F7,
* 132 * LIST F1 (FIX (FREQ(MAXVAL, 3))),
* 133 * LIST F2 (FIX (FREQ(MINVAL, 3))),
* 134 * IF FREQ(ZERO, 3) EQL -1 THEN
* 135 * LIST F3
* 136 * ENDIF,
* 137 * IF FREQ(MINPOS, 3) GRT 0 THEN
* 138 * LIST F4 (FREQ (FREQ(MINPOS, 3)))
* 139 * ENDIF,
* 140 * IF FREQ(MINNEG, 3) LES 0 THEN
* 141 * LIST F5 (FREQ (FREQ(MINNEG, 3)))
* 142 * ENDIF,
* 143 * ENDIF,
* 144 * IF FLAG(4) EQL 1 THEN
* 145 * LIST F7,
* 146 * LIST F6 (FIX(4)),
* 147 * LIST F7,
* 148 * LIST F1 (FIX (FREQ(MAXVAL, 4))),
* 149 * LIST F2 (FIX (FREQ(MINVAL, 4))),
* 150 * IF FREQ(ZERO, 4) EQL -1 THEN
* 151 * LIST F3
* 152 * ENDIF,
* 153 * IF FREQ(MINPOS, 4) GRT 0 THEN
* 154 * LIST F4 (FIX (FREQ(MINPOS, 4)))
* 155 * ENDIF,
* 156 * IF FREQ(MINNEG, 4) LES 0 THEN
* 157 * LIST F5 (FIX (FREQ(MINNEG, 4)))
* 158 * ENDIF,
* 159 * ENDIF,
* 160 * IF FLAG(5) EQL 1 THEN
* 161 * LIST F7,
* 162 * LIST F6 ( FIX(5) ),
* 163 * LIST F7,
* 164 * LIST F1 (FIX (FREQ(MAXVAL, 5))),
* 165 * LIST F2 (FIX (FREQ(MINVAL, 5))),
* 166 * IF FREQ(ZERO, 5) EQL -1 THEN
* 167 * LIST F3
* 168 * ENDIF,
* 169 * IF FREQ(MINPOS, 5) GRT 0 THEN
* 170 * LIST F4 (FIX (FREQ(MINPOS, 5)))

GBROOKS 28-MAR-1990 10:14:05.50 NSWC Dahlgren, Va.
DRS Rev 6.014 Created 27-OCT-1989 12:55 Page 4

* 171 * ENDIF,
* 172 * IF FREQ(MINNEG, 5) LES 0 THEN
* 173 * LIST F5 (FIX (FREQ(MINNEG, 5)))
* 174 * ENDIF,
* 175 * ENDIF;
* 176 *
* 177 * PRINT FULL SUMMARY
* 178 * USING MAXVAL, MINVAL, MINPOS, MINNEG, AMMAP$$;
% D R S - l , E n d o f F i l e l e n g t h
UDISK4:[GBROOKS.CLASS.SPING90.ENG5904.TEST]DATA_RANGE.DRS;

GBROOKS 28-MAR-1990 10:14:05.50 NSWC Dahlgren, Va.

SYMBOL REFERENCE

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>SYM NODE</th>
<th>SYM</th>
<th>ETS-STB</th>
<th>ETS</th>
<th>ASSOCIATED TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING</td>
<td>DEFINING</td>
<td>NAME</td>
<td>ADDR</td>
<td>TYPE</td>
<td>ADDR LENGTH</td>
</tr>
<tr>
<td>LINE</td>
<td>FILE SPECIFICATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Symbols in Block 1. Starting at Line 1 and Ending at Line 0
A C24E0 PDV E C24D0 1 AM(0)
AMOUNT C20E0 CTR BE8F4 43 AMOUNT
AMMAP$$ C23D8 MAP BCB58 300 Associative Memory
Mapping 2 TEST3 MULT PDV
C C2470 PDV E C2460 1 AM(2)
0 TEST3 MULT PDV
EVTMAP$$ C2400 MAP BCA28 76 Event Type Code
Distribution 1 TEST3 MULT PDV
EXEC C2360 SYNON C2336 1
130

<table>
<thead>
<tr>
<th></th>
<th>TEST3_MULT_PDV</th>
<th>FMT</th>
<th>BD008</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>C22C0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>C2298</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F3</td>
<td>C2270</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4</td>
<td>C2248</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F5</td>
<td>C2220</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F6</td>
<td>C21F8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F7</td>
<td>C21D0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXVAL</td>
<td>C21A8</td>
<td>MAP</td>
<td>BD184</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINNEG</td>
<td>C2130</td>
<td>MAP</td>
<td>BD9F4</td>
<td>300</td>
</tr>
<tr>
<td>Value</td>
<td>16</td>
<td>DATA_RANGE.DRS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINPOS</td>
<td>C2158</td>
<td>MAP</td>
<td>BD4E4</td>
<td>300</td>
</tr>
<tr>
<td>Value</td>
<td>15</td>
<td>DATA_RANGE.DRS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRS REPORTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

300 Maximum Value

Negative Near Zero

Positive Near Zero

300 Minimum Value

300 Zero Value Did Occur

PAGE F-13

<table>
<thead>
<tr>
<th></th>
<th>DATA_RANGE.DRS</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C21A8</td>
<td>MAP</td>
<td>BD184</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C2130</td>
<td>MAP</td>
<td>BD9F4</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C2158</td>
<td>MAP</td>
<td>BD4E4</td>
<td>300</td>
</tr>
</tbody>
</table>

Page F-13

***Synonym Stack Address***
C0288 Length 64 Used 0

***Item Encoding Stack Address***
C0388 Length 512 Used 60

***Symbol Table Address***
C0B88 Length 3048 Used 1684

***Event Tracking Storage Address***
BBFD0 Length 4096 Used 3797

Input Files Read
(0) UDISK4:[GBROOKS.CLASS.SPRING90.ENG5904.TEST]DATA_RANGE.DRS;
(1) NSWCROOT:[INC]TADPOLE_DRS_EXTEND;
(2) UDISK4:[GBROOKS.CLASS.SPRING90.ENG5904.TEST]TEST3_MULT_PDV;

EVENT OCCURRENCE TRACE
Page 6

18-MAR-1990 15:44:14.50

Session TEST3 Sequence 1
<table>
<thead>
<tr>
<th>TID</th>
<th>MID</th>
<th>TS</th>
<th>STATE</th>
<th>PID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RESULTS FOR AM: 0

Maximum Value : 1
Minimum Value  : 0

RESULTS FOR AM: 1

Maximum Value : 1
Minimum Value  : 1
Positive Near Zero Value : 1

RESULTS FOR AM: 2

Maximum Value : 2
Minimum Value  : -1
Positive Near Zero Value : 2
Negative Near Zero Value : -1

RESULTS FOR AM: 3

Maximum Value : 2
Minimum Value  : -3
Zero Value Did Occur :
Positive Near Zero Value : 1
Negative Near Zero Value : -1

Execution Summary (CPU Seconds)

<table>
<thead>
<tr>
<th>Module</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Initialization</td>
<td>0.012</td>
</tr>
<tr>
<td>User Interface</td>
<td>0.127</td>
</tr>
<tr>
<td>PDV Processing</td>
<td>0.063</td>
</tr>
<tr>
<td>Report Generation</td>
<td>0.015</td>
</tr>
<tr>
<td>Program Coverage</td>
<td>0.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Summary</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Summary Length</td>
<td>4229 Words</td>
</tr>
<tr>
<td>RPG Command File Length</td>
<td>51 Words</td>
</tr>
<tr>
<td>Buffered I-O Count</td>
<td>21</td>
</tr>
<tr>
<td>Direct I-O Count</td>
<td>51</td>
</tr>
<tr>
<td>Number of Page Faults</td>
<td>310</td>
</tr>
<tr>
<td>Dynamic Memory Utilization</td>
<td>107904 Words</td>
</tr>
<tr>
<td>Turn-around Time</td>
<td>00:00:06.04</td>
</tr>
</tbody>
</table>

Fatal Errors Were Not Encountered

Command Option Summary

/INPUT=UDISK4:[GBROCKS.CLASS.SPRING90.ENGR5904.TEST]DATA_RANGE.DRS;
/OUTPUT=UDISK4:[GBROCKS.CLASS.SPRING90.ENGR5904.TEST]TEST3.RPT;
/PDV=UDISK4:[GBROCKS.CLASS.SPRING90.ENGR5904.TEST]TEST3_MULT.PDV;
/MSGFIL=NSWCSROOT:[DAT]TADPOLE_DRS.DMF
/PSFOUT=UDISK4:[GBROOKS,CLASS,Spring90,ENGR5904.TEST]DATA_RANGE.PSF;
/WAS NOT Saved
/RPGCF=DRSRPGCF.DAT
/PLOT=UDISK4:[GBROOKS,CLASS,Spring90,ENGR5904.TEST]DATA_RANGE.CCDRAW;
/NOP$FIN
/NOSCHEMA
/EXTEND=NSWCSROOT:[INC]TADPOLE_DRS.EXTEND;
/NODDP
/SYM
/List

DRS REPORTS
REPORT FOR TEST PROGRAM 4
DRS Rev 6.014 Created 27-OCT-1989 12:55 Page 1

********** Reading PSF Created by DRS Rev 6.014 27-OCT-1989 12:55:03.58
* 1 \NOVPCMSG\*
* 2 NUMFLAGS = 256*
* 3 *
* 4 PRINT FULL SUMMARY*
* 5 USING TABLE, VALUE1, VAJIE2, VALUE3, VALUE4, AMMAP$ ;
* DRS - I, END of FILE
UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]TEST.DRS;1


SYMBOL REFERENCE

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>SYM NODE</th>
<th>SYM ETS-STD</th>
<th>ETS</th>
<th>ASSOCIATED TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAPPING</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRING</td>
<td>DEFINING</td>
<td>NAME ADDR</td>
<td>TYPE</td>
<td>ADDR LENGTH</td>
</tr>
<tr>
<td>LINE</td>
<td>FILE SPECIFICATION</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Symbols in Block 1. Starting at Line 0 and Ending at Line 0
A A C2BS8 PDVE C27F8 1 AM (0)
ACOUNT C28F8 MAP BD7B4 300 AM counter
AMMAP$ C2AB0 MAP BD230 300 Associative Memory
Mapping 0 TEST4.PSF
C C2B48 PDVE C27D8 1 AM (2)
EVTMAP$ C2AD8 MAP BD100 76 Event Type Code
Distribution 0 TEST4.PSF
EXEC C2A38 SYNON C2A10 1
F1 C2998 FMT BD6E0 16
F2 C2970 FMT BD720 15
F3 C2948 FMT BD75C 14
F4 C2920 FMT BD794 8
TABLE C2D0 MAP BDC64 300 Boolean values
TASK C2A86 SYNON C2A60 1
TBP_ID C29E8 SYNON C29C0 1
VALUE1 C28A8 MAP BE114 300 first value
VALUE2 C28A8 MAP BE5C4 300 second value
VALUE3 C28A8 MAP BEA74 300 third value
VALUE4  C2830  MAP    BEF24    300  fourth value
VALUE5  C2808  MAP    BF3D4    300  fifth value
X      C2B80  PDV E    C27E8 1  AM( 1)
Y      C2B10  PDV E    C27C8 1  AM( 3)

*******************************************************************************
*******************************************************************************
Synonym Stack Address    C0960 Length  64 Used  0
Item Encoding Stack Address C0A60 Length  512 Used 26
Symbol Table Address    C1260 Length  3048 Used 1664
Event Tracking Storage Address EC6A8 Length  4096 Used 3549

Input Files Read
  ( 0) UDISK4:[GBROOKS.CLASS.SPRING90.ENG95904.TEST]TEST.DRS;1
  ( 1) UDISK4:[GBROOKS.CLASS.SPRING90.ENG95904.TEST]TEST4.PSF;1
  ( 2) UDISK4:[GBROOKS.CLASS.SPRING90.ENG95904.TEST]TEST4.PDV;

EVENT OCCURRENCE TRACE
Page 3  Page F-16

DRS REPORTS

Session TEST4  Sequence 1  13-MAR-1990 19:46:56.00
TFCS Mode Software Version

ELAPSED TIME EVENT OCCURRENCE STATE PID
TID MID TS ASSOCIATED DATA

TRUTH TABLE
AM 0 AM 1 AM 2 AM 3
0 0 1 1
0 1 1 1

Execution Summary (CPU Seconds)

System Initialization 0.016
User Interface 0.062
PDV Processing 0.050
Report Generation 0.018
Program Coverage 0.000

Performance Summary Length 8057 Words 63 PRUs
RPG Command File Length 53 Words 1 PRUs
Buffered I-O Count 18
Direct I-O Count 49
Number of Page Faults 269
Dynamic Memory Utilization 61312 Words
Turn-around Time 00:00:07.90
Fatal Errors Were Not Encountered

Command Option Summary
/INPUT=UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]TEST.DRS;1
/OUTPUT=UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]TEST4.RPT;
/PDV=UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]TEST4.PDV;
/MSGFIL=NSWC$ROOT:[DAT]TADPOLE_DRS.DMF
/PSFOUT=UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]TEST4.PSF; Was
Saved
/RPGCF=DRSRPGCF.DAT
/PLOT=UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]TEST.CCDRAW;
/PSFIN=UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]TEST4.PSF; -1
/NOSCHMA
/EXTEND=NSWC$ROOT:[INC]TADPOLE_DRS.EXTEND;
/NODDP
/SYM
/LIST

DRS REPORTS
* 1 * \NOVPCMSG\  
* 2 * \NUMFLAGS = 256\  
* 3 *  
* 4 * FORMAT F1 ("Maximum Value : ", I'6');  
* 5 * FORMAT F2 ("Minimum Value : ", I'6');  
* 6 * FORMAT F3 ("Zero Value Did Occur : ", I'6');  
* 7 * FORMAT F4 ("Positive Near Zero Value : ", I'6');  
* 8 * FORMAT F5 ("Negative Near Zero Value : ", I'6');  
* 9 * FORMAT F6 ("RESULTS FOR AM: ", I'6');  
* 10 * FORMAT F7 (" ");  
* 11 *  
* 12 * MAP  
* 13 * MAXVAL (255) IS "Maximum Value",  
* 14 * MINVAL (255) IS "Minimum Value",  
* 15 * MINPOS (255) IS "Positive Near Zero Value",  
* 16 * MNNEG (255) IS "Negative Near Zero Value",  
* 17 * ZERO (255) IS "Zero Value Did Occur" ;  
* 18 *  
* 19 * /* OPERAND CAPTURE EVENT = 11 */  
* 20 * WHEN EVTYPE EQL 11,  
* 21 * /* SET AN INDICATION OF WHICH AM MATCH HAS OCCURRED */  
* 22 * SET FLAG (OCAVAIL) = 1,  
* 23 * /* TEST FOR MAX VAL */  
* 24 * IF FREQ (MAXVAL, OCAVAIL) EQL 0 THEN  
* 25 * /* SET INITIAL VALUE FOR THAT OPERAND */  
* 26 * MAP OCAVAIL INTO MAXVAL BY DATA,  
* 27 * ELSE  
* 28 * /* OCAVAIL IS WHICH AM YOU ARE DEALING WITH, I.E. 1,2,...ETC */  
* 29 * IF FREQ(MAXVAL,OCAVAIL) LES DATA THEN  
* 30 * /* ZERO THE OLD MAP VALUE OUT */  
* 31 * MAP OCAVAIL INTO MAXVAL BY - FREQ (MAXVAL,OCAVAIL),  
* 32 * /* PUT NEW DATA VALUE IN */  
* 33 * MAP OCAVAIL INTO MAXVAL BY DATA  
* 34 * ENDIF ,  
* 35 * ENDIF ,  
* 36 *  
* 37 * /* TEST FOR MIN VAL */  
* 38 * IF FREQ (MINVAL, OCAVAIL) EQL 0 THEN  
* 39 * MAP OCAVAIL INTO MINVAL BY DATA,  
* 40 * ELSE  
* 41 * IF FREQ(MINVAL,OCAVAIL) GRT DATA THEN  
* 42 * MAP OCAVAIL INTO MINVAL BY - FREQ (MINVAL,OCAVAIL),  
* 43 * MAP OCAVAIL INTO MINVAL BY DATA  
* 44 * ENDIF,  
* 45 * ENDIF,  
* 46 *  
* 47 * IF DATA EQL 0 THEN  
* 48 * MAP -1 INTO ZERO  
* 49 * ELSEIF DATA GRT 0 THEN  
* 50 * /* MAP MINIMUM VALUE IN */  
* 51 * IF FREQ (MINPOS,OCAVAIL) EQL 0 THEN
* 52 * MAP OCAVAIL INTO MINPOS BY DATA,
* 53 * ELSE
* 54 * IF FREQ (MINPOS, OCAVAIL) GRT DATA THEN
* 55 * MAP OCAVAIL INTO MINPOS BY - FREQ (MINPOS, OCAVAIL),
* 56 * MAP OCAVAIL INTO MINPOS BY DATA

GBROOKS 18-MAR-1990 16:10:02.43 NSWC Dahlgren, Va.
* 57 * ENDIF,
* 58 * ELSE /* CHECK FOR MINIMUM NEGATIVE VALUE */
* 59 * IF FREQ (MINNEG, OCAVAIL) EQL 0 THEN
* 60 * MAP OCAVAIL INTO MINNEG BY DATA,
* 61 * ELSE
* 62 * IF FREQ (MINNEG, OCAVAIL) LES DATA THEN
* 63 * MAP OCAVAIL INTO MINNEG BY - FREQ (MINNEG, OCAVAIL),
* 64 * MAP OCAVAIL INTO MINNEG BY DATA
* 65 * ENDIF,
* 66 * ENDIF;
* 67 * ENDIF;
* 68 * FINALLY,
* 70 * IF FLAG(0) EQL 1 THEN
* 71 * LIST F7,
* 72 * LIST F6 ( 0 ),
* 73 * LIST F7,
* 74 * LIST F1 (FIX (FREQ(MAXVAL, 0))),
* 75 * LIST F2 (FIX (FREQ(MINVAL, 0))),
* 76 * IF FREQ(ZERO, 0) EQL -1 THEN
* 77 * LIST F3
* 78 * ENDIF,
* 79 * IF FREQ(MINPOS, 0) GRT 0 THEN
* 80 * LIST F4 (FIX (FREQ(MINPOS, 0)))
* 81 * ENDIF,
* 82 * IF FREQ(MINNEG, 0) LES 0 THEN
* 83 * LIST F5 (FIX (FREQ(MINNEG, 0)))
* 84 * ENDIF,
* 85 * IF FLAG(1) EQL 1 THEN
* 86 * LIST F7,
* 87 * LIST F6 ( FIX (1) ),
* 88 * LIST F7,
* 89 * LIST F1 (FIX (FREQ(MAXVAL, 1))),
* 90 * LIST F2 (FIX (FREQ(MINVAL, 1))),
* 91 * IF FREQ(ZERO,1) EQL -1 THEN
* 92 * LIST F3
* 93 * ENDIF,
* 94 * IF FREQ(MINPOS, 1) GRT 0 THEN
* 95 * LIST F4 (FIX (FREQ(MINPOS, 1)))
* 96 * ENDIF,
* 97 * IF FREQ(MINNEG, 1) LES 0 THEN
* 98 * LIST F5 (FIX (FREQ(MINNEG, 1)))
* 99 * ENDIF,
* 103 * IF FLAG(2) EQL 1 THEN
* 104 * LIST F7,
* 105 * LIST F6 ( FIX(2) ),
* 106 * LIST F7,
* 107 * LIST F1 ( FIX (FREQ(MAXVAL, 2))),
* 108 * LIST F2 ( FIX (FREQ(MINVAL, 2))),
* 109 * IF FREQ(ZERO, 2) EQL -1 THEN
* 110 * LIST F3
* 111 * ENDF,
* 112 * IF FREQ(MINPOS, 2) GRT 0 THEN
* 113 * LIST F4 ( FIX (FREQ(MINPOS, 2)))

GBROOKS 18-MAR-1990 16:10:02.43 NSWC Dahlgren, Va.

* 114 * ENDF,
* 115 * IF FREQ(MINNEG, 2) LES 0 THEN
* 116 * LIST F5 ( FIX (FREQ(MINNEG, 2)))
* 117 * ENDF,
* 118 * ENDF,
* 119 * IF FLAG(3) EQL 1 THEN
* 120 * LIST F7,
* 121 * LIST F6 ( FIX (3) ),
* 122 * LIST F7,
* 123 * LIST F1 ( FIX (FREQ(MAXVAL, 3))),
* 124 * LIST F2 ( FIX (FREQ(MINVAL, 3))),
* 125 * IF FREQ(ZERO, 3) EQL -1 THEN
* 126 * LIST F3
* 127 * ENDF,
* 128 * IF FREQ(MINPOS, 3) GRT 0 THEN
* 129 * LIST F4 ( FIX (FREQ(MINPOS, 3)))
* 130 * ENDF,
* 131 * IF FREQ(MINNEG, 3) LES 0 THEN
* 132 * LIST F5 ( FIX (FREQ(MINNEG, 3)))
* 133 * ENDF,
* 134 * ENDF,
* 135 * IF FLAG(4) EQL 1 THEN
* 136 * LIST F7,
* 137 * LIST F6 ( FIX(4) ),
* 138 * LIST F7,
* 139 * LIST F1 ( FIX (FREQ(MAXVAL, 4))),
* 140 * LIST F2 ( FIX (FREQ(MINVAL, 4))),
* 141 * IF FREQ(ZERO, 4) EQL -1 THEN
* 142 * LIST F3
* 143 * ENDF,
* 144 * IF FREQ(MINPOS, 4) GRT 0 THEN
* 145 * LIST F4 ( FIX (FREQ(MINPOS, 4)))
* 146 * ENDF,
* 147 * IF FREQ(MINNEG, 4) LES 0 THEN
* 148 * LIST F5 ( FIX (FREQ(MINNEG, 4)))
* 149 * ENDF,
* 150 * ENDF,
* 151 * IF FLAG(5) EQL 1 THEN
* 152 * LIST F7,
* 153 * LIST F6 ( FIX(5) ),
LIST F7,
LIST F1 (FIX (FREQ(MAXVAL, 5))),
LIST F2 (FIX (FREQ(MINVAL, 5))),
IF FREQ(ZERO, 5) EQL -1 THEN
LIST F3
ENDIF,
IF FREQ(MINPOS, 5) GRT 0 THEN
LIST F4 (FIX (FREQ(MINPOS, 5)))
ENDIF,
IF FREQ(MINNEG, 5) LES 0 THEN
LIST F5 (FIX (FREQ(MINNEG, 5)))
ENDIF,
PRINT FULL SUMMARY
USING MAXVAL, MINVAL, MINPOS, MINNEG, AMMAP$;
UDISK4: [GBROOKS.CLASS.SPRING90.ENGR5904.TEST]DATA_RANGE.DRS;

GBROOKS 18-MAR-1990 16:10:02.43 NSWC Dahlgren, Va.
DRS Rev 6.014 Created 27-CCT-1989 12:55 Page 4

SYMBOL REFERENCE

MAP
SYMBOL SYM NODE SYM ETS-STB ETS ASSOCIATED TEXT
STRING DEFINING
NAME ADDR TYPE ADDR LENGTH
LINE FILE SPECIFICATION

******************************************************************************
Symbols in Block 1. Starting at Line 1 and Ending at Line 0
AMMAP$ C23A0 MAP BCD5B 300 Associative Memory
Mapping 2 TEST5.PDV
ETVMAP$ C23C8 MAP BCA28 76 Event Type Code
Distribution 1 TEST5.PDV
EXEC C2328 SYNON C2300 1
  5 TEST5.PDV
  F1 C2288 FMT BD008 15
    4 DATA_RANGE.DRS
  F2 C2260 FMT BD044 15
    5 DATA_RANGE.DRS
  F3 C2238 FMT BD080 15
    6 DATA_RANGE.DRS
  F4 C2210 FMT BD0BC 15
    7 DATA_RANGE.DRS
  F5 C21E8 FMT BD0F8 15
    8 DATA_RANGE.DRS
  F6 C21C0 FMT BD134 12
    9 DATA_RANGE.DRS
  F7 C2198 FMT BD164 8
    10 DATA_RANGE.DRS
  I C24E0 PDV E C24D0 1 AM(0)
    0 TEST5.PDV
MAXVAL C2170 MAP BD1B4 300 Maximum Value
    13 DATA_RANGE.DRS
MINNEG C20F8 MAP BDF94 300 Negative Near Zero
<table>
<thead>
<tr>
<th>Value</th>
<th>16</th>
<th>DATA_RANGE.DRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINPOS</td>
<td>C2120</td>
<td>MAP = BDAE4</td>
</tr>
<tr>
<td>Value</td>
<td>15</td>
<td>DATA_RANGE.DRS</td>
</tr>
<tr>
<td>MINVAL</td>
<td>C2148</td>
<td>MAP = BD634</td>
</tr>
<tr>
<td>NUM</td>
<td>C24A8</td>
<td>PDV E C2498</td>
</tr>
<tr>
<td>NUM0</td>
<td>C2438</td>
<td>PDV E C2428</td>
</tr>
<tr>
<td>NUM23</td>
<td>C2470</td>
<td>PDV E C2460</td>
</tr>
<tr>
<td>SUM</td>
<td>C2400</td>
<td>PDV E C23F0</td>
</tr>
<tr>
<td>TASK</td>
<td>C2378</td>
<td>SYNON C2350</td>
</tr>
<tr>
<td>TBP_ID</td>
<td>C22D8</td>
<td>SYNON C22B0</td>
</tr>
<tr>
<td>ZERO</td>
<td>C20D0</td>
<td>MAP = BE444</td>
</tr>
</tbody>
</table>

**Synonym Stack Address**
C0288 Length 64 Used 0
**Item Encoding Stack Address**
C0388 Length 512 Used 25
**Symbol Table Address**
C0B88 Length 3048 Used 1688
**Event Tracking Storage Address**
BBFD0 Length 4096 Used 3724

**DRS REPORTS**

**EVENT OCCURRENCE TRACE**

**Input Files Read**

(0) UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]DATA_RANGE.DRS;
(1) NSWC$ROOT:[INC]TADPOLE DRS.EXTEND;
(2) UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]TEST5.PDV;

**Session TEST5**
**Sequence** 1
**TFCS Mode**
**Software Version**

<table>
<thead>
<tr>
<th>TID</th>
<th>MID</th>
<th>TS</th>
<th>EVENT OCCURRENCE</th>
<th>STATE</th>
<th>PID</th>
</tr>
</thead>
</table>

**RESULTS FOR AM:**

| Maximum Value | : 25 |
| Minimum Value | : 1 |
| Positive Near Zero Value | : 1 |

**RESULTS FOR AM:**

| Maximum Value | : 24 |
| Minimum Value | : 1 |
Positive Near Zero Value : 1

RESULTS FOR AM:

Maximum Value : 23
Minimum Value : 23
Positive Near Zero Value : 23

RESULTS FOR AM:

Maximum Value : 0
Minimum Value : 0

Execution Summary (CPU Seconds)

<table>
<thead>
<tr>
<th>Activity</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Initialization</td>
<td>0.013</td>
</tr>
<tr>
<td>User Interface</td>
<td>0.125</td>
</tr>
<tr>
<td>PDV Processing</td>
<td>0.096</td>
</tr>
<tr>
<td>Report Generation</td>
<td>0.015</td>
</tr>
<tr>
<td>Program Coverage</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Performance Summary Length 8057 Words 63 PRUs
RPG Command File Length 51 Words 1 PRUs
Buffered I-O Count 19
Direct I-O Count 64
Number of Page Faults 296
Dynamic Memory Utilization 106880 Words
Turn-around Time 00:00:05.78

DRS REPORTS

Fatal Errors Were Not Encountered

Command Option Summary
/INPUT=UDISK4:[GBROOKS.CLASS.SPING90.ENG5904.TEST]DATA_RANGE.DRS;
/OUTPUT=UDISK4:[GBROOKS.CLASS.SPING90.ENG5904.TEST]TEST5.RPT;
/PDV=UDISK4:[GBROOKS.CLASS.SPING90.ENG5904.TEST]TEST5.PDV;
/MSGFIL=NSWC$ROOT:[DAT]TADPOLE_DRS.DMF
/PSFOUT=UDISK4:[GBROOKS.CLASS.SPING90.ENG5904.TEST]TEST5.PSF; Was
Saved
/RPGCF=DRS$RPGCF.DAT
/PLOT=UDISK4:[GBROOKS.CLASS.SPING90.ENG5904.TEST]DATA_RANGE.CCDRAW;
/NOPSFIN
/NOSCHEMA
/EXTEND=NSWC$ROOT:[INC]TADPOLE_DRS.EXTEND;
/NODDP
/SYM
/LIST

DRS REPORTS
REPORT FOR TEST PROGRAM 6

GBROOKS

DRS Rev 6.014 Created 27-OCT-1989 12:55 Page 1

* 1 * \NOVPCMSG\ 
* 2 * \LETS=6000\ 
* 3 * FORMAT F1 (" T R U T H T A B L E ") ; 
* 4 * FORMAT F2 (" (A OR C) AND (X AND Y) "); 
* 5 * FORMAT F3 (S'2',I'1',S'9',I'1',S'8',I'1',S'7',I'1') ; 
* 6 * FORMAT F4 (" "); 
* 7 * FORMAT F5 (" E AND F OR G "); 
* 8 * FORMAT F6 (" I OR K AND L OR M "); 
* 9 * FORMAT F7 (" (0 OR P) AND Q "); 
* 10 * FORMAT F8 (" B AND D AND H AND N "); 
* 11 * FORMAT F9 ("NUM(I) ANDB NUM(I+1) ORB NUM(I+2) "); 
* 12 * FORMAT F10 (S'2',I'1',S'13',I'1',S'15',I'1') ; 
* 13 * FORMAT F11 ("LOOP COUNTER I = ", I'3') ; 
* 14 * 
* 15 * MAP 
* 16 * AMOUNT (255) IS "AM counter "; 
* 17 * TABLE (255) IS "Boolean values"; 
* 18 * VALUE1 (255) IS "first value "; 
* 19 * VALUE2 (255) IS "second value "; 
* 20 * VALUE3 (255) IS "third value "; 
* 21 * VALUE4 (255) IS "fourth value "; 
* 22 * VALUE5 (255) IS "fifth value "; 
* 23 * VALUE6 (255) IS " "; 
* 24 * VALUE7 (255) IS " "; 
* 25 * VALUE8 (255) IS " "; 
* 26 * VALUE9 (255) IS " "; 
* 27 * 
* 28 * /* OPERAND CAPTURE EVENT = 11 */ 
* 29 * WHEN EVTYPE EQL 11, 
* 30 * IF DATA EQL 0 THEN 
* 31 * MAP OCAVAIL INTO TABLE BY DATA 
* 32 * ELSE 
* 33 * MAP OCAVAIL INTO TABLE BY -FREQ(TABLE,OCAVAIL), 
* 34 * MAP OCAVAIL INTO TABLE BY 1 
* 35 * ENDIF , 
* 36 * 
* 37 * MAP OCAVAIL INTO AMOUNT BY 1, 
* 38 * 
* 39 * /* ALLOWS UP TO 4 OCCURANCES OF THE SAME OPERAND CAPTURE */ 
* 40 * IF FREQ(AMOUNT, OCAVAIL) EQL 1 THEN 
* 41 * MAP OCAVAIL INTO VALUE1 BY FREQ(TABLE,OCAVAIL) 
* 42 * ELSEIF FREQ (AMOUNT, OCAVAIL) EQL 2 THEN 
* 43 * MAP OCAVAIL INTO VALUE2 BY FREQ(TABLE,OCAVAIL) 
* 44 * ELSEIF FREQ (AMOUNT, OCAVAIL) EQL 3 THEN 
* 45 * MAP OCAVAIL INTO VALUE3 BY FREQ(TABLE,OCAVAIL) 
* 46 * ELSEIF FREQ (AMOUNT, OCAVAIL) EQL 4 THEN 
* 47 * MAP OCAVAIL INTO VALUE4 BY FREQ(TABLE,OCAVAIL) 
* 48 * ELSEIF FREQ (AMOUNT, OCAVAIL) EQL 5 THEN 
* 49 * MAP OCAVAIL INTO VALUE5 BY FREQ(TABLE,OCAVAIL) 
* 50 * ELSEIF FREQ (AMOUNT, OCAVAIL) EQL 6 THEN 
* 51 * MAP OCAVAIL INTO VALUE6 BY FREQ(TABLE,OCAVAIL) 

DRS REPORTS Page F-24
* 52 * ELSEIF FREQ(AMCOUNT, OCAVAIL) EQ 7 THEN
* 53 * MAP OCAVAIL INTO VALUE7 BY FREQ(TABLE, OCAVAIL)
* 54 * ENSDF;
* 55 *
* 56 * FINALLY,


* 57 * LIST F4,
* 58 * LIST F1,
* 59 * LIST F2,
* 60 * /* (A OR C) AND (X AND Y) */
* 61 * IF FREQ(AMCOUNT, 0) GEQ 1 THEN
* 62 * LIST F3 (FIX (FREQ(VALUE1, 0)),
* 63 * FIX (FREQ(VALUE1, 1)),
* 64 * FIX (FREQ(VALUE1, 2)),
* 65 * FIX (FREQ(VALUE1, 3))
* 66 * ENSDF,
* 67 * IF FREQ(AMCOUNT, 0) GEQ 2 THEN
* 68 * LIST F3 (FIX (FREQ(VALUE2, 0)),
* 69 * FIX (FREQ(VALUE2, 1)),
* 70 * FIX (FREQ(VALUE2, 2)),
* 71 * FIX (FREQ(VALUE2, 3))
* 72 * ENSDF,
* 73 * IF FREQ(AMCOUNT, 0) GEQ 3 THEN
* 74 * LIST F3 (FIX (FREQ(VALUE3, 0)),
* 75 * FIX (FREQ(VALUE3, 1)),
* 76 * FIX (FREQ(VALUE3, 2)),
* 77 * FIX (FREQ(VALUE3, 3))
* 78 * ENSDF,
* 79 * IF FREQ(AMCOUNT, 0) GEQ 4 THEN
* 80 * LIST F3 (FIX (FREQ(VALUE4, 0)),
* 81 * FIX (FREQ(VALUE4, 1)),
* 82 * FIX (FREQ(VALUE4, 2)),
* 83 * FIX (FREQ(VALUE4, 3))
* 84 * ENSDF,
* 85 *
* 86 * /* E AND F OR G */
* 87 * LIST F4,
* 88 * LIST F1,
* 89 * LIST F5,
* 90 * IF FREQ(AMCOUNT, 4) GEQ 1 THEN
* 91 * LIST F3 (FIX (FREQ(VALUE1, 4)),
* 92 * FIX (FREQ(VALUE1, 5)),
* 93 * FIX (FREQ(VALUE1, 6))
* 94 * ENSDF,
* 95 * IF FREQ(AMCOUNT, 4) GEQ 2 THEN
* 96 * LIST F3 (FIX (FREQ(VALUE2, 4)),
* 97 * FIX (FREQ(VALUE2, 5)),
* 98 * FIX (FREQ(VALUE2, 6))
* 99 * ENSDF,
* 100 * IF FREQ(AMCOUNT, 4) GEQ 3 THEN
* 101 * LIST F3 (FIX (FREQ(VALUE3, 4)),
* 102 * FIX (FREQ(VALUE3, 5)),

DRS REPORTS Page F-25
* 103 *
  FIX (FREQ(VALUE3,6))
* 104 *
  ENDIF,
* 105 *
  IF FREQ(AMCOUNT,4) GEQ 4 THEN
* 106 *
  LIST F3 (FIX (FREQ(VALUE4,4)),
* 107 *
  FIX (FREQ(VALUE4,5)),
* 108 *
  FIX (FREQ(VALUE4,6)))
* 109 *
  ENDIF,
* 110 *
* 111 */ I OR K AND L OR M */
* 112 *
  LIST F4,
* 113 *
  LIST F1,

* 154 * IF FREQ(AMCOUNT,11) GEQ 3 THEN
* 155 * LIST F3 (FIX (FREQ(VALUE3,11))),
* 156 * FIX (FREQ(VALUE3,12)),
* 157 * FIX (FREQ(VALUE3,13)))
* 158 * ENDF,
* 159 * IF FREQ(AMCOUNT,11) GEQ 4 THEN
* 160 * LIST F3 (FIX (FREQ(VALUE4,11))),
* 161 * FIX (FREQ(VALUE4,12)),
* 162 * FIX (FREQ(VALUE4,13)))
* 163 * ENDF,
* 164 * /* B AND D AND H AND N */
* 165 * LIST F4,
* 166 * LIST F1,
* 167 * LIST F8,
* 168 * IF FREQ(AMCOUNT,14) GEQ 1 THEN
* 169 * LIST F3 (FIX (FREQ(VALUE1,14))),
* 170 * FIX (FREQ(VALUE1,15)),
* 171 * FIX (FREQ(VALUE1,16)),
* 172 * FIX (FREQ(VALUE1,17)))
* 173 * ENDF,
* 174 * IF FREQ(AMCOUNT,14) GEQ 2 THEN
* 175 * LIST F3 (FIX (FREQ(VALUE2,14))),
* 176 * FIX (FREQ(VALUE2,15)),
* 177 * FIX (FREQ(VALUE2,16)),
* 178 * FIX (FREQ(VALUE2,17)))
* 179 * ENDF,
* 180 * IF FREQ(AMCOUNT,14) GEQ 3 THEN
* 181 * LIST F3 (FIX (FREQ(VALUE3,14))),
* 182 * FIX (FREQ(VALUE3,15)),
* 183 * FIX (FREQ(VALUE3,16)),
* 184 * FIX (FREQ(VALUE3,17)))
* 185 * ENDF,
* 186 * IF FREQ(AMCOUNT,14) GEQ 4 THEN
* 187 * LIST F3 (FIX (FREQ(VALUE4,14))),
* 188 * FIX (FREQ(VALUE4,15)),
* 189 * FIX (FREQ(VALUE4,16)),
* 190 * FIX (FREQ(VALUE4,17)))
* 191 * ENDF,
* 192 * /* LOOP COUNTER I */
* 193 * IF FREQ(AMCOUNT,21) GRT 0 THEN
* 194 * LIST F4,
* 195 * LIST F4,
* 196 * LIST F11 (FIX(FREQ(VALUE1,21))),
* 197 * LIST F4
* 198 * ENDF,
* 199 * IF FREQ(AMCOUNT,21) GRT 1 THEN
* 200 * LIST F4,
* 201 * LIST F4,
LIST F11 (FIX(FREQ(VALUE2,21))),
LIST F4,
ENDIF,
IF FREQ(AMCOUNT,21) GRT 2 THEN
LIST F4,
LIST F4,
LIST F11 (FIX(FREQ(VALUE3,21))),
LIST F4
ENDIF,
IF FREQ(AMCOUNT,21) GRT 3 THEN
LIST F4,
LIST F4,
LIST F11 (FIX(FREQ(VALUE4,21))),
LIST F4
ENDIF,
IF FREQ(AMCOUNT,21) GRT 4 THEN
LIST F4,
LIST F4,
LIST F11 (FIX(FREQ(VALUE5,21))),
LIST F4
ENDIF,
ENDIF,
IF FREQ(AMCOUNT,21) GRT 5 THEN
LIST F4,
LIST F4,
LIST F11 (FIX(FREQ(VALUE6,21))),
LIST F4
ENDIF,
IF FREQ(AMCOUNT,21) GRT 6 THEN
LIST F4,
LIST F4,
LIST F11 (FIX(FREQ(VALUE7,21))),
LIST F4
ENDIF,
/* NUM(I) ANDB NUM(I+1) ORB NUM(I+2) */
LIST F4,
LIST F1,
LIST F9,
IF FREQ(AMCOUNT,18) GEQ 1 THEN
LIST F10 (FIX (FREQ(VALUE1,18))),
FIX (FREQ(VALUE1,19)),
FIX (FREQ(VALUE1,20))
ENDIF,
IF FREQ(AMCOUNT,18) GEQ 2 THEN
LIST F10 (FIX (FREQ(VALUE2,18))),


DRS REPORTS Page F-28
* 256 *       FIX (FREQ(VALUE2,19)),
* 257 *       FIX (FREQ(VALUE2,20))
* 258 *       ENDIF,
* 259 *       IF FREQ(AMCOUNT,18) GEQ 3 THEN
* 260 *       LIST F10 (FIX (FREQ(VALUE3,18))),
* 261 *       FIX (FREQ(VALUE3,19)),
* 262 *       FIX (FREQ(VALUE3,20)))
* 263 *       ENDIF,
* 264 *       IF FREQ(AMCOUNT,18) GEQ 4 THEN
* 265 *       LIST F10 (FIX (FREQ(VALUE4,18))),
* 266 *       FIX (FREQ(VALUE4,19)),
* 267 *       FIX (FREQ(VALUE4,20)))
* 268 *       ENDIF;
* 269 *
* 270 *       PRINT FULL SUMMARY ;
% D R S - I , E n d o f F i l e
UDISK4:[GBROOKS.CLASS.SPRING90.ENGR5904.TEST]COMBINATION_TEST6.DRS;


<table>
<thead>
<tr>
<th>MAPPING</th>
<th>SYMBOL</th>
<th>SYM NODE</th>
<th>SYM</th>
<th>ETS-STD</th>
<th>ETS</th>
<th>ASSOCIATED TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING</td>
<td>NAME</td>
<td>ADDR</td>
<td>TYPE</td>
<td>ADDR</td>
<td>LENGTH</td>
<td>(IF APPLICABLE)</td>
</tr>
<tr>
<td>LINE</td>
<td>FILE SPECIFICATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

***************************************************************************
***************************************************************************

Symbols in Block 1. Starting at Line 1 and Ending at Line 0
A
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
C42B0 PDV E C42A0 1 AM(0)
0 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
C
C4278 PDV E C4268 1 AM(1)
0 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
D
C3F68 PDV E C3F58 1 AM(15)
0 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
E
C41D0 PDV E C41C0 1 AM(4)
0 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
F
C3D78 SYNON C3D50 1
5 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
F1
C3CD8 PDV E C4188 1 AM(5)
0 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
F10
C3B70 PDV E C4188 1 AM(5)
0 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
F11
C3B48 PDV E C4188 1 AM(5)
0 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
F2
C3CB0 PDV E C4188 1 AM(5)
0 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
** Distribution **

** EXEC **
C3D78 SYNON C3D50 1
5 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
F1
C3CD8 PDV E C4188 1 AM(5)
0 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
F10
C3B70 PDV E C4188 1 AM(5)
0 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
F11
C3B48 PDV E C4188 1 AM(5)
0 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
F2
C3CB0 PDV E C4188 1 AM(5)
0 TEST6.PDV
AMCOUNT C3B20 MAP BCAD4 300 AM counter
16 COMBINATION_TEST6.DRS C3DF0 MAP BC3A8 300 Associative Memory
AMMAP$ Mapping 2 TEST6.PDV
B
Event Type Code
148

4  C3C88  COMBINATION_TEST6.DRS  FMT  BC2D8  14
5  C3C60  COMBINATION_TEST6.DRS  FMT  BC910  8
6  C3C38  COMBINATION_TEST6.DRS  FMT  BC930  16
7  C3C10  COMBINATION_TEST6.DRS  FMT  BC970  16
8  C3BE8  COMBINATION_TEST6.DRS  FMT  BC9B0  16
9  C3BC0  COMBINATION_TEST6.DRS  FMT  BC9F0  16
10  C3B98  COMBINATION_TEST6.DRS  FMT  BCA30  16
11  C4160  COMBINATION_TEST6.DRS  PDV E  C4150  1  AM(  6)
     0  TEST6.PDV
     0  PDV E  C3F20  1  AM( 16)
I  C4128  COMBINATION_TEST6.DRS  PDV E  C4118  1  AM(  7)
     0  TEST6.PDV
     0  PDV E  C40E0  1  AM(  8)
K  C40B8  COMBINATION_TEST6.DRS  PDV E  C40A8  1  AM(  9)
     0  TEST6.PDV
     0  PDV E  C4070  1  AM( 10)
M  C4080  COMBINATION_TEST6.DRS  PDV E  C3EE8  1  AM( 17)
     0  TEST6.PDV
     0  PDV E  C3BC0  1  AM( 18)
NUMI  C3E88  COMBINATION_TEST6.DRS  PDV E  C3E78  1  AM( 19)
     0  TEST6.PDV
NUMI2  C3E50  COMBINATION_TEST6.DRS  PDV E  C3E40  1  AM( 20)
     0  TEST6.PDV
     0  PDV E  C4038  1  AM( 11)
O  C4048  COMBINATION_TEST6.DRS  PDV E  C4000  1  AM( 12)
     0  TEST6.PDV
     0  PDV E  C3FC8  1  AM( 13)
P  C4010  COMBINATION_TEST6.DRS  PDV E  C3DA0  1
     0  TEST6.PDV
Q  C3FD8  COMBINATION_TEST6.DRS  PDV E  C3D00  1
     0  TEST6.PDV
TABLE  C3AF8  COMBINATION_TEST6.DRS  MAP  BCF84  300  Boolean values
17  COMBINATION_TEST6.DRS
TASK  C3D88  COMBINATION_TEST6.DRS  SYNON  C3DA0  1
     0  TEST6.PDV
TBP_ID  C3D28  COMBINATION_TEST6.DRS  SYNON  C3D00  1
     0  TEST6.PDV
VALUE1  C3AD0  COMBINATION_TEST6.DRS  MAP  BD434  300  first value
     0  TEST6.PDV
18  COMBINATION_TEST6.DRS
VALUE2  C3AA8  COMBINATION_TEST6.DRS  MAP  BD8E4  300  second value
     0  TEST6.PDV
19  COMBINATION_TEST6.DRS
VALUE3  C3A80  COMBINATION_TEST6.DRS  MAP  BDD94  300  third value
     0  TEST6.PDV
20  COMBINATION_TEST6.DRS
VALUE4  C3A58  COMBINATION_TEST6.DRS  MAP  BE244  300  fourth value
**SYMBOL REFERENCE**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>SYM NODE</th>
<th>SYM</th>
<th>ETS-STB</th>
<th>ETS</th>
<th>ASSOCIATED TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE5</td>
<td>C3A30 MAP</td>
<td>BE6F4</td>
<td>300</td>
<td>fifth value</td>
<td></td>
</tr>
<tr>
<td>VALUE6</td>
<td>C3A08 MAP</td>
<td>BEBA4</td>
<td>300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALUE7</td>
<td>C39E0 MAP</td>
<td>BF054</td>
<td>300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALUE8</td>
<td>C39B8 MAP</td>
<td>BF504</td>
<td>300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALUE9</td>
<td>C3990 MAP</td>
<td>BF9B4</td>
<td>300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>C4240 PDV E</td>
<td>C4230</td>
<td>1</td>
<td>AM( 2)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>TEST6.PDV</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>C4208 PDV E</td>
<td>C41F8</td>
<td>1</td>
<td>AM( 3)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>TEST6.PDV</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**UDISK4:** [GBROOKS.CLASS.SPRING90.MKGR5904.TEST]COMBINATION_TEST6.DRS;

(1) NSWC$ROOT:[INC]TADPOLE_DRS.EXTEND;
(2) UDISK4:[GBROOKS.CLASS5.SPRING90.ENG5904.TEST]TEST6.PDV;5

**EVENT OCCURRENCE TRACE**

```
Page 8

Session TEST6  Sequence  1  18-MAR-1990 15:27:14.50

<table>
<thead>
<tr>
<th>TID</th>
<th>MID</th>
<th>TS</th>
<th>EVENT OCCURRENCE</th>
<th>STATE</th>
<th>PID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TRUTH TABLE

(A AND C) AND (X AND Y)

0 0 0 0

TRUTH TABLE

E AND F OR G

0 0 0
```
TRUTH TABLE

I OR K AND L OR M
0 0 0 0

TRUTH TABLE
0 OR P) AND Q
0 1 0

TRUTH TABLE
B AND D AND H AND N
0 0 0 0

TRUTH TABLE
NUM(I) ANDB NUM(I+1) ORB NUM(I+2)
0 0 0 0
1 0 0 0
1 0 0 0
1 0 0 0

Execution Summary (CPU Seconds)

System Initialization 0.013
User Interface 0.198
PDV Processing 0.098
Report Generation 0.042
Program Coverage 0.000

Performance Summary Length 6133 Words 48 PRUs
RPG Command File Length 40 Words 1 PRUs
Buffered I-O Count 19
Direct I-O Count 45
Number of Page Faults 328
Dynamic Memory Utilization 120192 Words
Turn-around Time 00:00:06.18

Fatal Errors Were Not Encountered

Command Option Summary

/INPUT=UDISK4:[GBROOKS.CLASSSPRING90.ENGR5904.TEST]COMBINATION_TEST6.DRS;
/OUTPUT=UDISK4:[GBROOKS.CLASSSPRING90.ENGR5904.TEST]TEST6.RPT;
/PDV=UDISK4:[GBROOKS.CLASSSPRING90.ENGR5904.TEST]TEST6.PDV;5
/MSGFIL=NSWC$ROOT:[DAT]TADPOLE_DRS.DMF

/PSFOUT=UDISK4:[GBROOKS.CLASSSPRING90.ENGR5904.TEST]COMBINATION_TEST6.PSF; Was NOT Saved
/RPGCF=DRSRPGCF.DAT

/PLOT=UDISK4:[GBROOKS.CLASSSPRING90.ENGR5904.TEST]COMBINATION_TEST6.CCD
/NOPSFIN
/NOSSCHEMA
/EXTEND=NSWC$ROOT:[INC]TADPOLE_DRS.EXTEND;
/NODDP
/SYM
/LIST
APPENDIX G

PDL FOR MICRO-SEQUENCER

NAVAL SURFACE WEAPONS CENTER
DAHLGREN, VIRGINIA 22448

***************************
* 29 Mar 90
*
* PDL/81 V1.4(vc).403
*
* 5332-PD8
*
***************************
TABLE OF CONTENTS

Collect_data_coverage_information .............................. 2
Save_data_coverage_information ......................... 3
TRIDENT SGS
PAGE 2
29 Mar 90

Collect_data_coverge_information

REF

******************************************************************************
*     * 1 ... This procedure will look at each event sent from the PEI
*  2 ... interface. If the event is an AM pattern, processing
*  3 ... will take place if the operand capture bit is set.
*  4 ... Otherwise the next event will be
*  5 ... read.
*  6 BEGIN
*  7 DO WHILE no end of measurement event (event 31) and no
   fatal errors
*  8    read event type
*  9    IF event type is am pattern and
* 10    there was an AM set up for this address and
* 11    the AM had operand capture turned on then
* 12    ... Next event has to be the operand associated with
        this AM
* 13    Read next event
* 14    IF it is not an operand value event then
* 15    print error message
* 16    set fatal error indication to stop processing
* 17    ELSE
* 18    save AM address value
* 19    save operand value
* 20    set multiplexer select flag
set new-data register load flag
set data loaded signal
ENDIF
ENDIF
ENDDO
END

*******************************************************************************
Save_data_coverge_information

REF

*****************************************************************************

* 1 ... The procedure will save the AM address, operand value and
* 2 ... the pipeline information from the hardware controller.
* 3 ... This information will be used in the report generation
   process.
* 4 BEGIN
* 5 DO WHILE no end of measurement event (event 31) and no
   fatal errors
* 6 IF an AM event has occurred with the operand capture
   bit set then
* 7 write AM address and associated operand value to a
   file
* 8 wait for the read/write signal from the hardware
   compare logic
* 9 write pipeline information associated with this address
   to a file
* 10 ENDFI
* 11 ENDDO
* 12 END
*

*****************************************************************************
***************
*                *
* END OF DESIGN DOCUMENT *
*                *
***************

STATISTICS
----------

1457 lines in definition file(s)
48 lines in source file(s)

422 dictionary entries allocated
1340 ivb's allocated; 1243 in use.

43008 bytes of dynamic memory used
APPENDIX H

PDL FOR REPORT GENERATOR

NAVAL SURFACE WEAPONS CENTER
DAHLGREN, VIRGINIA  22448

******************************
*
*    29 Mar 90
*
* PDL/81 V1.4(vc).403
*
*    5332-PD8
*
******************************

158
TRIDENT SGS
PAGE 1.001
29 Mar 90 TABLE OF CONTENTS

TABLE OF CONTENTS
----------------

Combination_report ............................................ 2
Data_coverage .................................................... 3
Min_max_report ................................................... 4
Process_etu_init .................................................. 5
read_pdv .......................................................... 6
Report_generation ............................................... 7
Combination_report

REF

*****************************************************************************

* 1 ... This procedure will produce the combination report
* 2 ... for data coverage. This report indicates the combination
* 3 ... of data items that was produced at each high level boolean
* 4 ... statement.
* 5 BEGIN
* 6   Read list of combination statements specified by user
* 7   DO FOR each combination statement specified by user (128 max)
* 8   DO FOR each am match
* 9   IF this am match is for this combination statement
    then
*10   write value out as False (zero) or True (non-zero)
*11   ENDF
*12 ENDDO
*13 IF flag is set to compare these results to previous results
    then
*14   read in previous combinations found
*15   compare to current combination just found
*16   IF this is a new combination then
*17   write new combination found to output file
*18   ENDF
*19 ENDF
*20 write all combinations to output file and to file that will
   be saved -
* 21 for future runs
* 22 ENDDO
* 23 RETURN
* 24 END
*

******************************************************************************
TRIDENT SGS
PAGE 3
29 Mar 90

Data_coverage

REF

******************************************************************************************

* 1 ... This procedure is the main controller for the data
* 2 ... coverage report generation. It will read the user's
* 3 ... input. If all parameters are valid, control is passed
* 4 ... to Read_pdv to process the information.
* 5 BEGIN
* 6    Read user input parameters
* 7    Open PDV file specified by user
* 8    IF PDV file does not exist then
* 9        Print error
* 10       Set fatal error condition
* 11      ELSE
* 12        IF user specified a previous file to be used for
                   combinational results then
* 13           Open previous file
* 14           IF the file does not exist then
* 15                Print error
* 16                Set fatal error condition
* 17           ELSE
* 18            IF user wants data coverage min/max report then
* 19                set flag indicating to perform min/max report
generation
* 20            ENDIF
* 21            IF user wants combinational data coverage report
then
* 22  set flag indicating to perform combinational report generation
* 23  ENDF
6 * 24  CALL Read_pdv
* 25  ENDF
* 26  ENDF
* 27  ENDF
* 28  END
*

*****************************************
Min_max_report

REF

******************************************************************************
*
* 1 ... This procedure will produce the minimum and maximum
* 2 ... report for data coverage. It will write the results to
* 3 ... the output file and to a save file that can be used for
* 4 ... future runs to compare to new data.
* 5 BEGIN
* 6 ... User may specify a previous file to be compared with this run
* 7 ... for new minimums and maximums. This is used for cumulative
* 8 ... information
* 9 IF previous file flag is set then
*10   compare each old minimum and maximum to new data collected
*11   IF old minimum < new minimum then
*12     save minimum as old minimum
*13   ELSE
*14     save minimum as new minimum
*15   ENDIF
*16   IF old maximum > new maximum then
*17     save maximum as old maximum
*18   ELSE
*19     save maximum as new maximum
*20   ENDIF
*21   IF zero value occurred in old or new data then
* 22        set flag that zero value occurred
* 23        ENDIF
* 24        Find closest value to zero (both positive and negative)
            between -
* 25        old and new data and save
* 26        ELSE
* 27        use information from new data collected for report
generation
* 28        ENDIF
* 29        DO FOR each am requested for output
* 30        write maximum value to output file
* 31        write minimum value to output file
* 32        write indication of zero value occurring or not occurring
* 33        write closest positive value to zero
* 34        write closest negative value to zero
* 35        save new statistics for cumulative reporting
* 36        ENDDO
* 37        RETURN
* 38        END

***************************************************************************
TRIDENT SGS
PAGE 5
29 Mar 90

Process_etu_init

REF

P A G E

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

* 1 ... This procedure will read the ETU initialization block from the
* 2 ... PDV file. It will set up an array of values which correspond
* 3 ... to each possible AM match for easy access by the processing
* 4 ... procedures
* 5 BEGIN
* 6 DO FOR each of the 255 possible am matches
* 7 IF operand capture bit is set then
* 8 store associated tid value
* 9 store associated mid value
*10 IF amc is equal to 7 then
*11 store address as virtual operand address
*12 ELSEIF amc is equal to 6 then
*13 store address as physical operand address
*14 ELSEIF amc is equal to 5 then
*15 store address as virtual program address
*16 ELSEIF amc is equal to 4 then
*17 store address as physical program address
*18 ENDIF
*19 ELSE
*20 ... we are not interested in this am match
*21 store a don't care value for each of the fields
*22 ENDIF
* 23    ENDDO
* 24    RETURN
* 25    END

*****************************************************************************************
TRIDENT SGS
PAGE 6
29 Mar 90

read_pdv

REF

*****************************************************************************
* 1 ... This procedure reads the different blocks in the
* 2 ... PDV file and either skips the block, or processes it,
* 3 ... depending on the block id.
* 4 BEGIN
* 5   Read Session Block (block 1)
* 6   Save session information for report
* 7   Read ETU initialization Block
5* 8   CALL Process_ebu_init (to process information in block 2)
* 9   DO WHILE block is not PMD data block or end of file reached
* 10   Skip corresponding number of words
* 11   Read block id
* 12   ENDDO
* 13   IF end of file reached then
* 14   ... Fatal error, do not continue processing
* 15   print error "no PMD data block"
* 16   ELSE
7* 17   CALL Report_generation (to print final report)
* 18   ENDF
* 19   RETURN
* 20   END
*
*****************************************************************************
Report_generation

REF

PEND

******
* 1 ... This procedure will figure out what type of report
* 2 ... to produce. It will then call the appropriate report
* 3 ... generation procedure along with an indication of
* 4 ... whether to look at a previous generated file to compare
* 5 ... results.
* 6 BEGIN
* 7 IF user specified to produce a cumulative report by looking
    at
* 8     previous results then
* 9     Verify that the file the user specified containing the
    previous
* 10     results exist.
* 11     Set flag indicating to look at data from previous results
    file
* 12     specified by the user.
* 13     ENDIF
* 14 IF user wants the min/max report then
4* 15     CALL min_max_report
* 16     ENDIF
* 17 IF user wants the combinational report then
2* 18     CALL combination_report
* 19     ENDIF
* 20 RETURN
* 21 END
STATISTICS

--------

1457 lines in definition file(s)
176 lines in source file(s)

428 dictionary entries allocated
1337 ivb's allocated; 1249 in use.

43008 bytes of dynamic memory used
APPENDIX I
LIFE CYCLE COST ANALYSIS
## LIFE CYCLE COST ANALYSIS

### Research and Development Cost Summary

<table>
<thead>
<tr>
<th>Program Activity</th>
<th>Cost Category</th>
<th>Cost by Program Year</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Designator</td>
<td>Year 1</td>
<td>Year 2</td>
</tr>
<tr>
<td>Alternative A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Customer Costs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. System/product management</td>
<td>Crm</td>
<td>12,250</td>
<td>13,475</td>
</tr>
<tr>
<td>2. Production planning</td>
<td>Crp</td>
<td>3,467</td>
<td>---</td>
</tr>
<tr>
<td>B. Supplier Costs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. System/product mgmt.</td>
<td>Crm</td>
<td>8,948</td>
<td>9,842</td>
</tr>
<tr>
<td>2. Product planning</td>
<td>Crp</td>
<td>2,168</td>
<td>2,384</td>
</tr>
<tr>
<td>3. Engineering design</td>
<td>Cre</td>
<td>10,000</td>
<td>11,000</td>
</tr>
<tr>
<td>4. Design Documentation</td>
<td>Crd</td>
<td>1,800</td>
<td>1,980</td>
</tr>
<tr>
<td>5. System Test and Evaluation</td>
<td>Crt</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Total Research and Development Cost</td>
<td>Cr</td>
<td>38,633</td>
<td>38,681</td>
</tr>
<tr>
<td>Alternative A</td>
<td>Cr</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Production/Construction Cost Summary

<table>
<thead>
<tr>
<th>Program Activity</th>
<th>Cost Category</th>
<th>Cost by Program Year</th>
<th>Total Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Designator</td>
<td>Year 2</td>
<td>Year 3</td>
</tr>
<tr>
<td>Alternative A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Customer Costs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. System/product management</td>
<td>Crm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B. Supplier Costs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Industrial eng. and operational analysis</td>
<td>Cpi</td>
<td>3,000</td>
<td>3,300</td>
</tr>
<tr>
<td>2. Manufacturing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a. Recurring</td>
<td>Cpmr</td>
<td>16,300</td>
<td>36,980</td>
</tr>
<tr>
<td>b. Non-recurring</td>
<td>Cpmn</td>
<td>9,760</td>
<td>---</td>
</tr>
<tr>
<td>3. Quality Control</td>
<td>Cpq</td>
<td>7,768</td>
<td>8,544</td>
</tr>
<tr>
<td>4. Initial Logistics support</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a. Supply support</td>
<td>Cpls</td>
<td>75,000</td>
<td>120,000</td>
</tr>
<tr>
<td>b. Test and equipment</td>
<td>Cplt</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>c. Technical data</td>
<td>Cpld</td>
<td>1,200</td>
<td>---</td>
</tr>
<tr>
<td>d. Personnel training</td>
<td>Cplp</td>
<td>3,500</td>
<td>---</td>
</tr>
<tr>
<td>Total Production and Construction cost for Alternative A</td>
<td>Cp</td>
<td>116,528</td>
<td>168,824</td>
</tr>
</tbody>
</table>
### Operation and Support Cost Summary

<table>
<thead>
<tr>
<th>Program Activity</th>
<th>Cost Category</th>
<th>Cost by Program Year</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternative A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Operating Personnel</td>
<td>Coop</td>
<td>---</td>
<td>---</td>
<td>2450</td>
<td>49010</td>
<td>49010</td>
<td>49010</td>
<td>49010</td>
</tr>
<tr>
<td>2. Transportation</td>
<td>Cott</td>
<td>---</td>
<td>---</td>
<td>8000</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>3. Unscheduled Maintenance</td>
<td>Coim</td>
<td>---</td>
<td>---</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
</tr>
<tr>
<td>4. Maintenance Facilities</td>
<td>Coim</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>5. Scheduled Maintenance</td>
<td>Coim</td>
<td>---</td>
<td>---</td>
<td>1508</td>
<td>3016</td>
<td>3016</td>
<td>3016</td>
<td>3016</td>
</tr>
<tr>
<td>7. Test and Support Equipment</td>
<td>Coim</td>
<td>---</td>
<td>---</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Total for Alternative A</td>
<td>Coim</td>
<td>---</td>
<td>---</td>
<td>34531</td>
<td>52544</td>
<td>52544</td>
<td>52544</td>
<td>52544</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program Activity</th>
<th>Cost Category</th>
<th>Cost by Program Year</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternative A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Operating Personnel</td>
<td>Coop</td>
<td>49010</td>
<td>49010</td>
<td>49010</td>
<td>49010</td>
<td>49010</td>
<td>49010</td>
<td>465595</td>
<td></td>
</tr>
<tr>
<td>2. Transportation</td>
<td>Cott</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>3. Unscheduled Maintenance</td>
<td>Coim</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>2180</td>
<td></td>
</tr>
<tr>
<td>4. Maintenance Facilities</td>
<td>Coim</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>5. Scheduled Maintenance</td>
<td>Coim</td>
<td>3016</td>
<td>3016</td>
<td>3016</td>
<td>3016</td>
<td>3016</td>
<td>3016</td>
<td>28652</td>
<td></td>
</tr>
<tr>
<td>7. Test and Support Equipment</td>
<td>Coim</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>Total for Alternative A</td>
<td>Coim</td>
<td>52544</td>
<td>52544</td>
<td>52544</td>
<td>52544</td>
<td>52544</td>
<td>52544</td>
<td>507427</td>
<td></td>
</tr>
<tr>
<td>Program Activity</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Alternative A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Organizational maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a. Total labor hours</td>
<td>---</td>
<td>---</td>
<td>13.52</td>
<td>13.52</td>
<td>13.52</td>
<td>13.52</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b. Personnel Cost</td>
<td>---</td>
<td>---</td>
<td>196</td>
<td>196</td>
<td>196</td>
<td>196</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c. Material Handling</td>
<td>---</td>
<td>---</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d. Documentation Cost</td>
<td>---</td>
<td>---</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Unscheduled Maintenance Cost</td>
<td>---</td>
<td>---</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unscheduled Maintenance Cost

<table>
<thead>
<tr>
<th>Program Activity</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Alternative A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Organizational maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a. Total labor hours</td>
<td>13.52</td>
<td>13.52</td>
<td>13.52</td>
<td>13.52</td>
<td>13.52</td>
<td>13.52</td>
</tr>
<tr>
<td>b. Personnel Cost</td>
<td>196</td>
<td>196</td>
<td>196</td>
<td>196</td>
<td>196</td>
<td>196</td>
</tr>
<tr>
<td>c. Material Handling</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>d. Documentation Cost</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Total Unscheduled Maintenance Cost</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
</tr>
</tbody>
</table>
### Life Cycle Cost Breakdown

<table>
<thead>
<tr>
<th>Cost Category</th>
<th>Alternative A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cost</td>
</tr>
<tr>
<td>1. Research and Development</td>
<td></td>
</tr>
<tr>
<td>a. System/Product Mgmt Crm</td>
<td>70,164</td>
</tr>
<tr>
<td>b. Product Planning Crp</td>
<td>8,019</td>
</tr>
<tr>
<td>c. Engineering Design Crc</td>
<td>32,100</td>
</tr>
<tr>
<td>d. Design Documentation Crd</td>
<td>5,958</td>
</tr>
<tr>
<td>e. System Test and Eval Ctd</td>
<td>18,430</td>
</tr>
<tr>
<td>Subtotal</td>
<td>135,671</td>
</tr>
<tr>
<td>2. Production and Construction Cpl</td>
<td></td>
</tr>
<tr>
<td>a. Industrial engineering and</td>
<td></td>
</tr>
<tr>
<td>operations analysis Cpl</td>
<td>6,300</td>
</tr>
<tr>
<td>b. Manufacturing - recurring Cpmr</td>
<td>53,280</td>
</tr>
<tr>
<td>c. Manufacturing - nonrecurring Cpmr</td>
<td>9,760</td>
</tr>
<tr>
<td>d. Quality Control Cplq</td>
<td>16,312</td>
</tr>
<tr>
<td>e. Initial Logistics Support Cplq</td>
<td></td>
</tr>
<tr>
<td>1. Supply Support Cplq</td>
<td>195,000</td>
</tr>
<tr>
<td>2. Test and Support</td>
<td>---</td>
</tr>
<tr>
<td>3. Technical data Cpld</td>
<td>1,200</td>
</tr>
<tr>
<td>4. Personnel Training Cplq</td>
<td>3,500</td>
</tr>
<tr>
<td>Subtotal</td>
<td>285,332</td>
</tr>
<tr>
<td>3. Operation and Support Cost Crtd</td>
<td></td>
</tr>
<tr>
<td>a. Operating Personnel Coop</td>
<td>465,595</td>
</tr>
<tr>
<td>b. Distribution - Transportation Ctd</td>
<td>8,000</td>
</tr>
<tr>
<td>c. Unscheduled Maintenance Cols</td>
<td>2,180</td>
</tr>
<tr>
<td>d. Maintenance Facilities Cols</td>
<td>---</td>
</tr>
<tr>
<td>e. Scheduled Maintenance Cols</td>
<td>28,652</td>
</tr>
<tr>
<td>f. Maintenance Personnel</td>
<td>2,000</td>
</tr>
<tr>
<td>g. Training Cols</td>
<td></td>
</tr>
<tr>
<td>1. Test and Support</td>
<td>1,000</td>
</tr>
<tr>
<td>2. Equipment Cols</td>
<td></td>
</tr>
<tr>
<td>Subtotal</td>
<td>507,427</td>
</tr>
<tr>
<td>Grand Total</td>
<td>828,450</td>
</tr>
</tbody>
</table>
Present Cost Evaluation at Interest Rate of 10%

<table>
<thead>
<tr>
<th>Category</th>
<th>Year</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>R &amp; D</td>
<td></td>
<td>36633</td>
<td>38681</td>
<td>58357</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Production</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Construction</td>
<td></td>
<td>---</td>
<td>116528</td>
<td>168824</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Operation /</td>
<td></td>
<td>---</td>
<td>34531</td>
<td></td>
<td>51036</td>
<td>51036</td>
<td>51036</td>
</tr>
<tr>
<td>Support</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>36633</td>
<td>165209</td>
<td>261712</td>
<td>51036</td>
<td>51036</td>
<td>51036</td>
</tr>
<tr>
<td>P/F, 10, n</td>
<td></td>
<td>35121</td>
<td>128280</td>
<td>196624</td>
<td>34857</td>
<td>31688</td>
<td>28809</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Year</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>R &amp; D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Production</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Construction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation /</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support</td>
<td></td>
<td>51036</td>
<td>51036</td>
<td>51036</td>
<td>51036</td>
<td>51036</td>
<td>51036</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>51036</td>
<td>51036</td>
<td>51036</td>
<td>51036</td>
<td>51036</td>
<td>51036</td>
</tr>
<tr>
<td>P/F, 10, n</td>
<td></td>
<td>26191</td>
<td>23808</td>
<td>21644</td>
<td>19679</td>
<td>17888</td>
<td>16260</td>
</tr>
<tr>
<td>Grand Total: P/F = 580842</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


## Appendix J

### ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>Associative Memory</td>
</tr>
<tr>
<td>AMC</td>
<td>Arithmetic Magnitude Comparator</td>
</tr>
<tr>
<td>AMML</td>
<td>Associative Memory and Match Logic</td>
</tr>
<tr>
<td>DCU</td>
<td>Data Coverage Unit</td>
</tr>
<tr>
<td>DRS</td>
<td>Data Reduction Software</td>
</tr>
<tr>
<td>ETU</td>
<td>Event Trace Unit</td>
</tr>
<tr>
<td>HC</td>
<td>Hardware Controller</td>
</tr>
<tr>
<td>IPCU</td>
<td>Interpretive Program Coverage Unit</td>
</tr>
<tr>
<td>MR</td>
<td>Main Ram</td>
</tr>
<tr>
<td>NDR</td>
<td>New Data Register</td>
</tr>
<tr>
<td>PDL</td>
<td>Program Design Language</td>
</tr>
<tr>
<td>PDV</td>
<td>Performance Data Volume</td>
</tr>
<tr>
<td>PEI</td>
<td>Performance Evaluation Interface</td>
</tr>
<tr>
<td>PSW</td>
<td>Program Status Word</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>SDS</td>
<td>Software Development System</td>
</tr>
<tr>
<td>TS</td>
<td>Time Stamp</td>
</tr>
<tr>
<td>VEST</td>
<td>Verification and Evaluation System</td>
</tr>
</tbody>
</table>